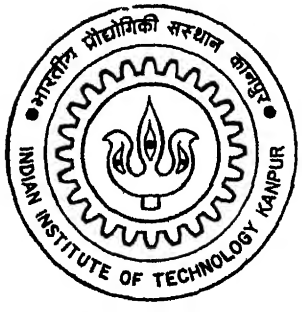
OBJECT - ORIENTED SIMULATION OF FLEXIBLE MANUFACTURING SYSTEM FOR COMPARISON OF DISPATCHING RULES

by

SANTOSH KUMAR SAXENA



DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

APRIL, 1996

OBJECT-ORIENTED SIMULATION OF FLEXIBLE MANUFACTURING SYSTEM FOR COMPARISON OF DISPATCHING RULES

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

by
SANTOSH KUMAR SAXENA

to the
DEPARTMENT OF INDUSTRIAL & MANAGEMENT ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
APRIL, 1996

IME-1996-M-SAX-OBJ

CERTIFICATE

This is to certify that the present work on "Object-oriented simulation of flexible manufacturing system for comparison of dispatching rules", by Santosh Kumar Saxena has been carried out under my supervision and has not been submitted elsewhere for the award of a degree

17/4/96

April, 1996

(Kripa Shanker)

Professor

Industrial & Management Engineering Deptt

Indian Institute of Technology

Kanpur - 208016

ACKNOWLEDGMENT

First and foremost, I would like to place on record, my deepest gratitude to Dr Kripa Shanker for his immense help and guidance throughout this thesis tenure and outside it His broad vision for both human values and technical knowledge will always inspire and guide me

Special thanks are due to Mr Neeraj Kumar and Mr Amit Rathore for their help in my thesis work The time I spent in IIT K with Kansal, Gopal, Jhaji, Murari, Pushpesh, Gopal, Negi, Kapil, Puneet, Rishi, Mittal, Parvesh, Shobhit, and Capt Tiwari will be amongst the most cherished period of my life, especially the birthday and job parties in CR and G Sir Without them, life would not have been that great in this institute

I take this opportunity to thank all the faculty members for making my stay in IIT K, an intellectually invigorating experience I also express my gratitude towards whole IME family

(Santosh Kumar Saxena)

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The concepts of object-oriented programming (OOP) are having a profound impact on computer software development The advantages offered by OOP philosophy in the area of manufacturing systems simulation appear to be very attractive The basic concepts of OOP viz abstraction, encapsulation, inheritance, information hiding, and polymorphism render software systems more understandable, modifiable, and reusable In the present work, a comprehensive object-oriented simulation model for an FMS is developed for testing of various dispatching rules 'C++' is used as the object-oriented programming language because it joins together two separate programming paradigms - the procedural language tradition represented by C, and the object-oriented language tradition, represented by the enhancements C++ adds to C

The FMS configuration consists of a set of multifunctional machining centers with input and output queues, loading and unloading stations Parts of various types arrive at the system in a Poisson manner with a known arrival rate The parts may have technological precedence relationship among themselves They go through an ordered sequence of operations which can be performed on more than one machining centers with stochastic processing times The dispatching rules considered in the present study are the ones which are reported to perform well in job shop environment viz (i) FIFO, (ii) WINQ, (iii) EDD, (iv) AT, and (v) SIO These dispatching rules are tested for varying degrees of flexibility and precedence The performance criteria used for comparing these rules are (i) Average machine utilization, (ii) Mean flow time, (iii) Mean tardiness, and (iv) Mean waiting time

Among the dispatching rules considered, SIO generally appears to perform better than other rules for reducing the flow time and tardiness based objectives, particularly at medium and high flexibility and precedence levels The other rule, which does well for part based objectives, is WINQ Due date based rule, EDD, offer reduced tardiness at high precedence levels Average machine utilization tend to increase with increasing precedence at various degrees of flexibility for all rules All the part related performance measures

generally increase with increasing levels of precedence at fixed flexibility and decrease with increasing flexibility at fixed precedence for all rules

The present system is implemented in Borland C++ Version 3 1

CHAPTER ONE

INTRODUCTION

Flexible Manufacturing Systems (FMS) attempt to achieve flexibility and productivity simultaneously in order to meet the demands of today's competitive markets They have gone through rapid evolution and development and many effective systems have been installed in a variety of configurations in a wide range of manufacturing environment This growing interest in FMS has brought many new problems and issues These problems and some of their solutions are shared at the International Conferences on Flexible Manufacturing Systems held annually since 1982 Several national and international journals and magazines cover many FMS aspects Papers covering FMS topics are presented at almost every conference on manufacturing , automation etc

There is not a standard accepted definition for the general term 'flexible manufacturing system' Many definitions have been given For example , Byrkett et al (1988) state that

A flexible manufacturing system (FMS)is a manufacturing system in which groups of numerically controlled machines (machine centers) and a material handling system work together under computer control

Other definitions are based on the capability or performance of a 'typical' FMS For example, Kaltwasser et al (1986) state that

Flexible manufacturing systems (FMS) are highly automated production systems, able to produce a great variety of different parts by using the same equipment and the same control system

FMS is a general term for a broad collection of production systems which may take several different structural forms Maccarthy et al (1993) defined FMS as

A flexible manufacturing system is a production system capable of producing a variety of part types, which consists of CNC or NC machine tools connected by an

automated material handling system The operation of the whole system is under computer control

The aim of flexible manufacturing system is to achieve the efficiency of automated mass production while retaining the flexibility of a job shop

1.1 STRUCTURAL TAXONOMY OF FMS

In a broad sense, an FMS can consist of three subsystems fabrication, machining, and assembly Each of these subsystems may be highly integrated with automated storage and a computer aided design system Andrew Kusiak (1985) presented a structural approach to FMS (see Fig 1 1)

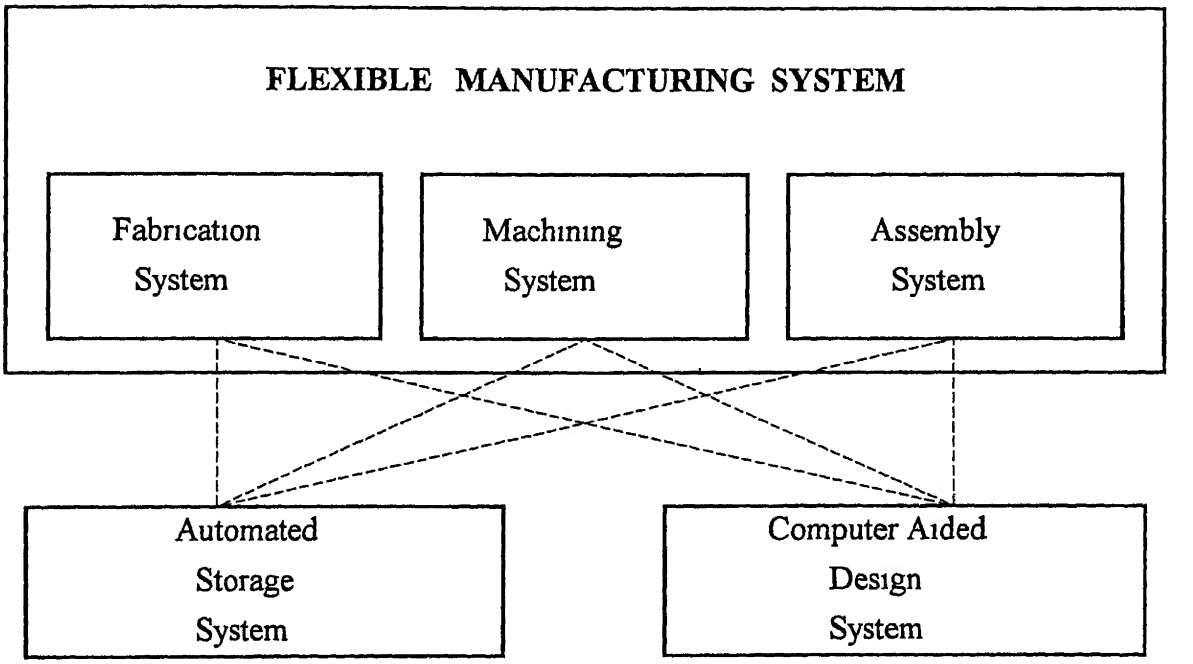


Figure 1.1 : Integration of flexible manufacturing system with related systems

Integration between an FMS and an automated storage system is very often a material handling system (for example an automated guided vehicle system) and a

computer system The computer aided design system is integrated with an FMS through it's strong design and manufacturing links

Based on the number of numerically controlled (NC) machines and their arrangement FMSs can be divided into the following classes

(1) flexible manufacturing module (Dupont - Gatelmand 1981)

(2) flexible manufacturing cell

(3) flexible manufacturing group (Traub 1983)

(4) flexible production system (Dupont - Gatelmand 1981)

(5) flexible manufacturing line

An approximate relationship between the number of different parts per system per year and the annual production rate for parts for these five classes of FMSs is shown in

Fig 1 2

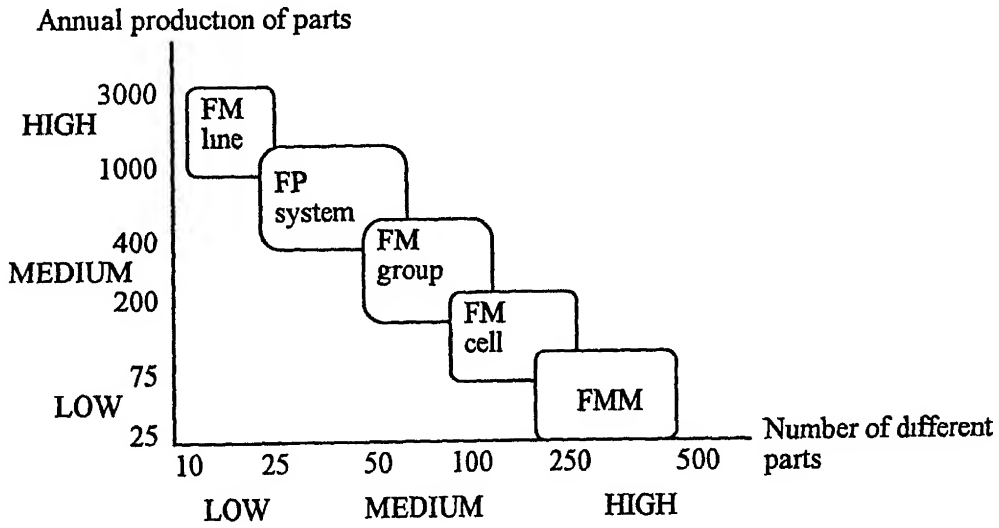


Figure 1.2 : Relationships of different classes of FMSs, numbers of different parts and annual production of parts

A flexible manufacturing module (FMM) can be defined as an NC machine augmented by a part buffer, a tool changer, a pallet changer etc The last two items can be replaced by a robot

A flexible manufacturing cell (FMC) consists of several FMMs and can be built according to product or process type (Warnecke and Steinhilper 1982) A flexible manufacturing group (FMG) is a collection of FMCs and FMMs in the same manufacturing area joined by a material handling system under common computer control

A flexible production system (FPS) consists of FMGs representing different manufacturing areas, fabrication, machining and assembly A flexible manufacturing line (FML) can be defined as a set of dedicated machine tools

A widely recognized classification of FMSs characterize them as being 'dedicated' or 'random'

A dedicated FMS employs a set of general or special purpose machines, an automated material handling system, part specific fixtures and magazines with a fixed set of tools on each machine It produces a small family of processing requirements (Denzler and Boe 1987)

A random FMS employs a set of general purpose machines, an automated material handling system, modular pallet fixtures and an automated tool loading system It is capable of producing a large family of widely differing parts (Denzler and Boe 1987)

1.2 FLEXIBILITY IN FMS

Flexibility is an essential feature of FMS It can be viewed in many different ways Flexibility is defined as the ability of a manufacturing system to cope with changing circumstances (Buzacott and Mandelbaum 1985) or instability caused by the environment (Mascarenhas 1981) It is one of the key objectives of any manufacturing system and a critical measure of total manufacturing performance (Chatterjee et al 1984) It ensures that the manufacturing can be both cost efficient and customized at

the same time As set up time decreases, small batch production can be as economical as large scale manufacturing This enables the organization to change its competitive strategy from economies of scale to economies of scope

More importantly, flexibility embodies competitive value for a manufacturer In conditions of environmental scarcity , organizations with inflexible technologies are the most vulnerable because of the limits on range and variety of products and the high cost of operating below capacity

Environmental conditions under which the flexibility may be required and/or justified and operating requirements under which the associated benefits can be realized are yet not clearly identified (Jaikumar 1986) In fact, most of the research attention has been devoted to the hardware requirements of flexibility with little concern for the system management and control mechanisms that are necessary for it's realization

Various types of flexibilities have been discussed by Browne et al (1984) , Carter (1986) and several other authors from time to time

1.2.1 CLASSIFICATION OF MANUFACTURING FLEXIBILITIES

Machine flexibility The ease of making changes required to produce a given set of part types It can be measured by the time to replace defective tools, the time to change tools in a tool magazine, and the time assemble or mount new fixtures

Process flexibility The ability to produce a given set of part types, each possibly using different materials, in several ways It can be measured by the number of part types that can be processed simultaneously without using batches

Product flexibility The ability to change over to produce a new (set of) product(s), economically and quickly It can be measured by the time required to switch from one part mix to another, not necessarily of the same part types

Routing flexibility The ability to handle breakdowns and to continue producing the given set of part types It can be measured by the robustness of the FMS when breakdowns occur, that is, the production rate does not decrease dramatically and parts continue to be processed

Volume flexibility The ability to operate an FMS profitably at different production volumes It can be measured by the minimum volumes of all part types for which the system can be run profitably

Expansion flexibility The capability of building a system and expanding it as needed, easily and modularly It can be measured by the maximum possible size of the FMS

Operation flexibility The ability to interchange the ordering of several operations for each part type

Production flexibility The universe of part types that the FMS can produce It can be measured by the level of existing technology Production flexibility is dependent on all the other flexibility types

Most of the flexibilities in FMS arise primarily because of the versatile nature of resources and possibility of alternative substitutes As an example, consider an FMS that has six machines in operation These machines will accumulate some idle times during operation This could happen when a machine is waiting to process a job that is currently being operated upon by another machine in the system In another system configuration, the FMS could have eight machines in operation These machines will also accumulate some idle times during operation These idle times will be different from those generated in the previous configuration A statistically significant difference would indicate an increase or decrease in the mean idle times If

the latter configuration results in higher idle times, it would be safe to infer that an increase in the number of machines has caused a reduction in the system's 'volume flexibility' Using the definitions of Browne et al (1984), volume flexibility is the ability to operate an FMS profitably at different production volumes If by adding machines, the total machine idle time increases, then the FMS would not be operating as profitably as in the former configuration

Less obvious may be the impact of a change in the number of machines on the 'product flexibility' and the 'machine flexibility' A greater number of machines would enhance the ability of the system to accommodate these changes This would reduce the waiting times for the jobs thus enhancing machine flexibility A lower value of the waiting times would supplement the system's capability to changeover to produce a new part, thus improving product flexibility Another possibility is an increase in the number of routes that the jobs can take through the machines Increasing the number of routes increases the routing flexibility of the system An increase in number of routes may also allow for more effective utilization of the machines thus affecting other flexibility types

1.3 FMS DECISION PROBLEMS

While an FMS possesses the attractive combination of automation and flexibility, the production management problems are rather more complex The various management problems that need to be addressed at different stages of an FMS's life cycle have been discussed by Stecke (1983) Jaikumar found that FMS yields the flexibility of job shops with the efficiency of mass production systems if used properly It is not the hardware but the system's management that is to be blamed for poor FMS performance To exploit the potential of FMSs, it is not enough to know the operational problems associated with them but also to know how susceptible the performance is to changes in operating policies and demand variations

In an FMS, two groups of problems are of particular importance (1) design and (2) operational The first group is concerned with the optimal selection of all the FMS components, and the second with their optimal utilization

1.3.1 DESIGN PROBLEMS

To design an FMS, the following problems should be solved

(1) Organizational problem, that is selection of part families to be manufactured

(2) Selection of an FMS production system

(3) Selection of a material handling system

(4) Selection of fixtures and pallets

(5) Selection of an appropriate computer system

(6) Layout and integration of all the above systems

1.3.2 PLANNING AND SCHEDULING PROBLEMS

Because FMSs have a high capital cost , a high rate of utilization is necessary to ensure short return on investment

FMS planning problem includes the determination of the parts to be simultaneously machined, grouping of machines, allocation of pallets and fixtures to part types and loading of machines FMS scheduling problem includes determining the optimal input sequence of parts, and an optimal sequence at each machine for a given part mix

Stecke has identified five interrelated production planning problems which must be solved prior to system operation

(1) part type selection

(2) machine grouping

(3) production ratio determination

(4) resource allocation, and

(5) loading

The production ratio problem is to determine the relative ratio at which the part type selected for manufacturing will be produced The resource allocation problem concerns with allocating the limited resources of pallets and fixtures among the selected part types

1.3.3 LOADING PROBLEM

Assuming that part type selection, machine grouping, production ratio and resource allocation problems have been solved, the loading problem is specified as selecting a subset of jobs from the job pool, and assigning their operations to the appropriate machines in the ensuing planning period so as to achieve certain specified objectives while meeting the system constraints The loading problem is to allocate the operations and the associate tools required for the selected set of part types among the machine groups, subject to the technological and capacity constraints of the manufacturing system

Stecke(1983) has described six objectives of loading in FMS

(1) balancing the machine processing times,

(2) minimizing the number of movements,

(3) balancing the workload per machine for a system of groups of pooled machines of equal sizes,

(4) unbalancing the workload per machine for a system of groups of pooled machines of unequal sizes,

(5) filling the tool magazines as densely as possible, and

(6) maximizing the sum of operation priorities

Stecke and Solberg (1981) identified a set of necessary constraints associated with the loading problem

(1) each required operation and all associated tools must be assigned to at least one machine,

(2) an operation can be assigned only to the machines that are capable of performing it,

(3) the tools required for the entire set of operations, assigned to any machine, must not exceed the capacity of the tool magazine for that machine

When deciding on which part to load next, it is necessary to have (i) an order for that part, (ii) a pallet capable of holding it, (iii) the necessary raw material or WIP part, and (iv) an open slot at the load center If a set of parts satisfies the above conditions, then a scheduler chooses to use either an 'order file oriented' part loading rule or a 'systems oriented' part loading rule (Denzler et al 1987) An order file oriented part loading rule selects the part to be loaded next by considering only the characteristics of the parts in the FMS It assumes that the flexibility of the FMS will adequately accommodate any sequence of parts that have been loaded One such widely used rule is to load that part next which has the shortest processing time (SPT) A systems oriented part loading rule selects the next part so as to improve the utilization of the FMS (Denzler et al 1987) An example of such a rule is to assign the operations to the machines so as to minimize the inter-machine part movements (Stecke and Solberg 1981)

1.3.4 SCHEDULING PROBLEM

Subsequent to loading, the next step is running the FMS in real time This involves scheduling the operations according to some dispatching rule, as the machines become available FMSs pose unique and challenging problems with regard to the real time scheduling of workparts For example, according to Chang et al (1986)

(1) After a part is loaded, which machines should it visit and at what time should it be processed on specific machines ?

(2) If a machine fails, how should the schedule be adjusted for each part in the system ?

(3) If a high priority part is loaded, how should the schedules be adjusted for each part so as to accommodate this high priority part ?

Dispatching rules or heuristics can be classified as 'local' or 'global' A local dispatching rule only requires information about the job completing for service at a

particular machine center A global dispatching rule requires additional information about the jobs or machine status at other machine centers or waiting lines (Gere 1966, Gupta et al 1989)

Local dispatching rules are easily implemented, but they fail to take advantage of the global information that is available in an FMS environment An example of a local dispatching rule is FIFO (first in first out) According to this rule, jobs are scheduled in the same sequence as they arrive Global dispatching rules capitalize on the information about various machining centers in the FMS However, due to the complex data requirements, global rules are extremely difficult to implement (Chang et al (1986) A popular global dispatching rule is WINQ (work in next queue) This rule selects the job that will go on to its next operation at the machine with the least workload

1.4 AN OVERVIEW OF PAST RESEARCH ON FMS OPERATIONAL PROBLEMS

The FMS loading problem has been rigorously pursued in the research literature Bernardo and Mohamed (1992) have shown that loading decisions affect FMS flexibility A widely used loading objective is the balancing of machine workloads The rationale is that if the workloads are uniform, there will be less congestion and system performance will be improved However, Stecke (1981) points out that the practice of balancing may be too restrictive since the flexibility of the machines may not be fully utilized

Although balancing of workloads is the most applied objective, a survey result of 19 FMS installations in the U S by Smith et al (1986) casts a totally different perspective on this objective Industry considers meeting due dates to be the most important Maximization of system utilization, and minimization of in process inventory are considered to be next most important In the literature, various objectives related to makespan, resource utilization, due dates, etc have been

considered for the scheduling problem (Wassenhove and Gelders, 1990 , Deckro et al , 1982 , Ghiassi, 1984 , Norbis and Smith, 1988)

Various loading objectives and corresponding relevant research articles are categorized in Table 1 1 In Table 1 1, the loading objectives are listed in descending order of importance as found by Smith et al (1986) The classification is based on the FMS environment which considers the length of the planning period and the type of demand Four combinations are identified as (1) infinite period and static demand, (2) finite period and static demand, (3) infinite period and variable demand, (4) finite period and variable demand Most of the researchers assume infinite period and static demand

A study on job shop loading is given by Irastorza and Deane (1974) It is concerned with selecting new jobs to be released to the shop during each production period so that job completion dates are as close as possible to their due dates

Nof et al (1979) have reviewed the loading problem in FMS as a composite of product mix and process selection problems

A simulation study was carried out by Stecke and Solberg (1981) for dedicated type FMS examining five loading strategies versus sixteen dispatching rules As their study is based on a dedicated type of FMS, the loading strategies are simply the procedures to allocate operations to a number of machines of same type The study does not consider due dates of the jobs

Stecke (1983) gave the first mathematical formulation for the FMS loading problem The grouping and loading are formulated as non linear 0 - 1 mixed integer problem Solution methodologies with several computational simplifications have been developed

Shanker and Tzen (1985) considered the loading problem in random FMS with the bi criterion objective of balancing workload amongst the machining centres and meeting the due dates of the jobs Heuristic procedures were suggested and a mathematical

model was developed Effects of loading on system performance under different dispatching rules were then examined on a simulation model

Kimemia and Stanley (1985) used network flow optimization approach to solve optimal part routing problem

Barrada and Stecke solved the loading problem using a branch and bound approach with balancing of workload on all machines as the objective

Shanker and Srinivasulu (1989) developed a two stage branch and backtrack procedure with the objective of maximizing the assigned workload

Mukhopadhyay et al (1991) formulated the scheduling problem as a hierarchical process and solved it through eigen vector analysis of priority ordering

Chandra et al (1991) proposed an 'Intelligent' dispatching rule (EXPERT) for FMS that takes into account the coming states and trends of the system

Mukhopadhyay et al (1992) developed a heuristic procedure for the loading problems in FMS on the concept of essentiality ratio for the objective of minimizing the system unbalance and thereby maximizing throughput They applied their algorithm to the problem of Shanker and Srinivasulu (1989) and observed that the proposed heuristic leads to a better solution in terms of system unbalance and throughput

Arizono et al (1992) solved the scheduling problem for minimizing the total actual flow time by using Gaussian machine model which is one of the stochastic neural network models

Udyan and Christy (1992) investigated the effect of machine flexibility and number of part families on system performance Their methodology was a combination of Computer Simulation, Response Surface Methodology and the Hooke Jeeves procedure

Kim (1993) examined the possibility of using surrogate objectives in the FMS loading problem Empirical research was done for the relationships using a good solution for given workload assignments It was found that surrogate objective for minimizing mean tardiness is more effective when due dates of orders are tighter

Kim and Jano (1994) presented a due date based approach for part type selection in FMSs The approach was based on the idea of using information from the solution of an approximate aggregate scheduling problem as the basis for determining release priorities Several different dispatching rules were evaluated Results indicated that the above approach performed far better than using simpler procedures

Benjaafar (1994) examined the effect of varying flexibility in either part production requirements or the machine capabilities

Modi and Shanker (1994) addressed a generalized loading problem with the objective of part movement minimization in an FMS

Raghu and Rajendran (1995) suggested an efficient scheduling and due date setting policy using three different methodologies based on a search algorithm, simulated annealing algorithm and a combination of simulated annealing algorithm and regression analysis

Many heuristics have been suggested for loading and dispatching rules In addition to development of heuristics specific to the FMS environment, several dispatching rules that are tested to be performing good in a general job shop environment, are tested by various researchers in FMS environment

Montazerri and Wassenhove (1990) have analysed the performance of a number of scheduling rules for an FMS

The popular Shortest Processing Time (SPT) rule arranges jobs on the basis of the smallest imminent processing time Due date based dispatching rules are aimed at reducing the mean tardiness of the jobs A number of approaches of setting job and operation due dates are proposed , most popular of which is the Total Work Content (TWK) method (Conway 1965) It assigns the due dates of the jobs according to the following definition

$$D = A + kP$$

where D = due date of the job

A = arrival time of the job

P = total processing time of the job

k = parameter specified by the management

Caroll developed the complex COVERT dispatching rule which also combines job slack and SPT COVERT rule prioritizes jobs according to the largest ratio of expected incremental delay cost for an operation to the imminent operation time

Schultz proposed CEXSPT rule which applies the SPT rule in a controlled manner using due date information to expedite jobs that are late or behind schedule

Anderson and Nyirendra (1990) proposed two new rules combining the SPT rule, the Critical Ratio rule and the Slack per remaining work rule The two new rules were effective in minimizing the mean tardiness and the number of tardy jobs

1.5 PRESENT WORK

The above discussion highlights the need of dispatching rules for the allocation of scarce resources The dispatching rules can have a large impact on performance measures in automated systems, largely because all components of such systems are tightly interconnected In order to optimize the use of scarce resources like machines, and meet the targets of due dates etc , the need of a comprehensive simulation model is strongly felt which can study the various interactions among the dispatching rules

The environment of the FMS considered for the present work consists of workstations which are the real processing stations for the jobs that arrive at the system according to some arrival process These workstations consist of one or more machines but they accept only one job at a time for processing Each workstation has a limited number of input buffer spaces from which parts are drawn for processing Upon completion of processing, parts are placed in a limited capacity output buffer There is a loading station at which jobs accumulate before they enter the FMS and an unloading station where they collect after completing all the operations The jobs that undergo same sequence of operations are assumed to belong to same type (family) These jobs may have technological precedence relationship among themselves i e some parts are required to complete all their operations so that their successor can enter the system This is a very practical consideration missing in most of the research

work done on FMS simulation The jobs go through an ordered sequence of operations Each of these operations is associated with a set of workstations on which it can be performed with each of the workstations possibly having a different processing time When a machine is required to process jobs from different families, a setup is required This is because some or all of the tooling may need to be changed A number of decisions have to be taken while running the FMS These include

(i) selection of jobs to enter the FMS

(ii) selection of the job to be loaded on the machine for processing

(iii) selection of the workstation among various stations vying for the next operation of the job

(iv) whether or not to enter the part at the loading station

Some dispatching rules, which are reported to perform well in a general job shop kind of environment, and their sensitivity with varying degree of flexibility and precedence are studied The performance measures related both to workstations and jobs are considered

1.6 ORGANIZATION OF THE THESIS

Chapter II gives an idea about the *'Object Oriented Programming paradigm'* After describing the important features of OOP, it outlines the advantages of object oriented modeling of manufacturing systems and its limitations

Chapter III deals with the system's modeling and the methodology used A brief justification is also provided for using simulation as the medium for modeling and analysis The description of the system considered for this study is also presented Dispatching rules considered and suggested are also discussed in this chapter

Chapter IV deals with the implementation aspects of the system A brief justification on the use of OOP and C++ as programming media instead of some special or general purpose simulation language is given at the beginning Various classes designed for the FMS and their member functions are also briefly described

Chapter V deals with the system validation Here the system considered for the experimentation is described and then various experiments have been mentioned The results of various experiments have been presented

Chapter VI presents the conclusions arrived at after analyzing the output of various experiments and suggestions for further work

Table 1.1 Various loading objectives and corresponding relevant research articles

FMS ENVIRONMENT				
LOADING POLICIES	**INFINITE PERIOD, STATIC DEMAND**	**FINITE PERIOD , STATIC DEMAND**	**INFINITE PERIOD, VAR. DEMAND**	**FINITE PERIOD, VAR. DEMAND**
Due date satisfaction		Shanker and Tzen (1985) Shanker and Srinivasulu(1989)		
Min in process inventory		Chakravarty and Shtub (1984)		Bernardo and Mohamed (1992)
Maximize production rate	Stecke(1981), Stecke and Solberg(1985)			Bernardo and Mohamed (1992)
Minimize setup and tool change/ cost	Rajagopalan (1986)	Kiran and Tansel (1986), Kouvelis and Lee(1991)		Hwang and Shogan (1989)
Minimize flow time	Green and Sadowski (1986)			
Balance machine use	Stecke (1983), Whitney and Goul (1986), Co Biermann and Chen (1990), Stecke and Morin(1985)			Bernardo and Mohamed (1991)

CHAPTER TWO

OBJECT ORIENTED METHODOLOGY

2.1 OBJECT ORIENTED PROGRAMMING

The concepts of Object Oriented Programming (OOP) are having a profound impact on computer software construction Different programming paradigms have evolved over time to handle increasingly complex problems to be solved by computers Modular programming was the first step that required large programs to be broken down into smaller manageable components This concept got the primary support from the invention of subroutines Structured programming concept refined the idea further by enforcing functional decomposition and making the static nature of program very near to its runtime nature (Goto less program or non sphagetti code)

The term 'Object Oriented Technology' encompasses a body of methods, processes and tools used to develop software systems from objects Object oriented technology has spread far beyond its original simulation language (Simula in 70s) but programming with objects still retains the spirit of real world simulation

Object oriented programming (OOP) requires one to view the world as a collection of discrete objects which act and react in a common environment by exchanging messages The two key elements of an object oriented programming paradigm are object and message An object is some private data / attributes and a set of behavior that it is capable of The private data is to represent the object's status, and the behavior of an object is defined by a collection of procedures, each known as method / member function A class is group of objects which have the same characteristics It is that software module which provides a complete definition of the capabilities of members of the class Capabilities are either provided by the procedures and data storage contained within the immediate class definition or inherited from other class definitions to which this class is related OOP embodies following concepts which result in making software systems more understandable, modifiable and reusable

2 1 1 ***Abstraction :***

Abstraction denotes the essential characteristics of an object that distinguish it from all other objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer All the operations that can be meaningfully performed on an object and the way object reacts determines the behavior of the object Abstraction helps in formalizing the external nature of the object both as client (requiring services) and as a server (providing services) and hence serves to separate the essential behavior from the implementation Abstraction has both dynamic and static properties The abstraction may also vary depending according to the viewer and thus same object may have different abstraction depending upon the service required by the clients

2 1 2 ***Encapsulation :***

Encapsulation means that an object's data and procedures are enclosed within a tight boundary, one which cannot be penetrated by other objects Data stored within an object is directly accessible only by the procedures that have been defined as part of the class to which the object belongs It essentially hides the representation of an object and implementation of its methods The protection provided by encapsulation is two ways protecting the internal states of the object from outside (client) objects and prevent dependence of client objects on internal representation of the (server) object Encapsulation restricts any undesired side effects from changing the contents of any object's data Since all the data and procedures related to an object are located in one place, changes to be made are confined to one location

2 1 3 ***Message passing :***

Message passing is a necessary result of encapsulation In order for one object to affect the internal condition of another object, the first object must request (by sending a message) the second object to execute one of its procedures Objects communicate with each other by sending messages When an object receives a message, it

activates a corresponding method A message consists of three parts the name of receiving object, the name of the method that server knows how to handle and any number of parameters that are required by that method (optional) Messages allow for two way communication between communicating objects The server object can return a value or an object in response to a request for service from a client object

2 1 4 ***Inheritance :***

Inheritance provides for a low level form of software reuse The concept of inheritance promotes software reusability and helps to eliminate redundancy in coding OOP classes are defined in a hierarchical tree structure Each class in the tree structure inherits the methods and data storage structure of all its superclasses Inheritance allows the construction of new objects from existing objects by extending, reducing or otherwise modifying their functionality Inheritance is an important mechanism for representing 'IS A' and 'PART OF' relationship hierarchy consistently across analysis, design, and implementation phase

2.2 OOP VERSUS STRUCTURED PROGRAMMING

Object oriented programming is a newer way of programming than the traditional structured technique Structured programming has been associated exclusively with functional design methods such as functional decomposition design, data flow design and data structure design In OOP, objects are first categorized in classes and organized hierarchically according to the dependency and similarity The consequent classes of objects have greater functional and feature cohesion Then relationships between classes, such as inheritance, are designed, often from the natural domain dependent relations between classes found during domain analysis

Advantages of OOP over traditional (procedural) programming have been documented by Cox and Meyer According to Meyer " object oriented design may be defined as a technique which, unlike classical (functional) design, bases the modular decomposition of a software system on the classes of the objects the system manipulates,

not on the functions the system performs " Meyer goes on to point out that objects remain more or less stable, whereas functions tend to adapt to changing needs or circumstances

In terms of simulation modeling requirements, following the object oriented paradigm has the important advantage of preserving the bulk of developed code for general use in model building Each model building exercise then performs the particular functions that are of interest at that time The object definitions remain independent of the functions of the system being modeled The characteristics of OOP allow us to rethink our entire approach to system's modeling using computers

More detailed discussion on the relevance of OOP concepts to modeling of manufacturing systems can be found in Adiga (1989) Object oriented simulation languages / systems are described by King and Fischer (1986), Larkin et al (1988) and Lomow and Baezner (1989) A detailed comparison of conventional and object oriented approaches to simulation was done by Nazmi and Stein (1989)

2.3 OBJECT ORIENTED MODELING OF MANUFACTURING SYSTEMS

The concepts underlying OOP can be extended to simulation modeling The performance of a manufacturing system is highly influenced by control policies used in its operation In evaluating the system performance, it is desirable not only to consider the physical components and the physical configuration, but also the results of different operating policies A complex hierarchical decision making structure exists in a manufacturing system The decisions are based on available information, which is often incomplete, inaccurate and delayed A decision maker at each level uses heuristics, personal expertise, company rules, and policies to arrive at a control decision Traditional system modeling tools do not provide convenient structures for specifying these interactions

There are two reasons why a modeler should incorporate explicit and separate information processing and decision making structures into his / her models

(1) to facilitate a higher degree of reusability, and

(2) to facilitate a more 'natural' model building environment

Traditional simulation languages do not provide natural constructs for separately and distinctly modeling physical elements, information flows and control decisions

There are several areas in the context of manufacturing systems where an object oriented modeling technology can be gainfully employed Manufacturing systems are frequently studied in order to make either long term strategic decisions (manufacturing system design) or short term tactical decisions (production control and scheduling) Object oriented models of manufacturing systems offer some distinct advantages over traditional modeling practices Table 2 1 contains a summary of several factors on which the traditional modeling paradigm is compared to the object oriented paradigm

A system consists of a number of components with some form of interrelationship Specifically in a manufacturing system these components could be machines, raw materials, finished parts, material handling devices, human, information documents, computers, etc Software models of manufacturing systems capture the relationships or interactions between these components In object oriented modeling, each component of the manufacturing system is abstracted as an object A typical object oriented manufacturing system model consists of workstation objects, work flow item objects, conveyer objects, bill of material objects, etc Unlike traditional modeling techniques, object oriented modeling creates models that exhibit close resemblance to the real world systems For example Figure 2 1 depicts a machine object abstracted from a real world machine

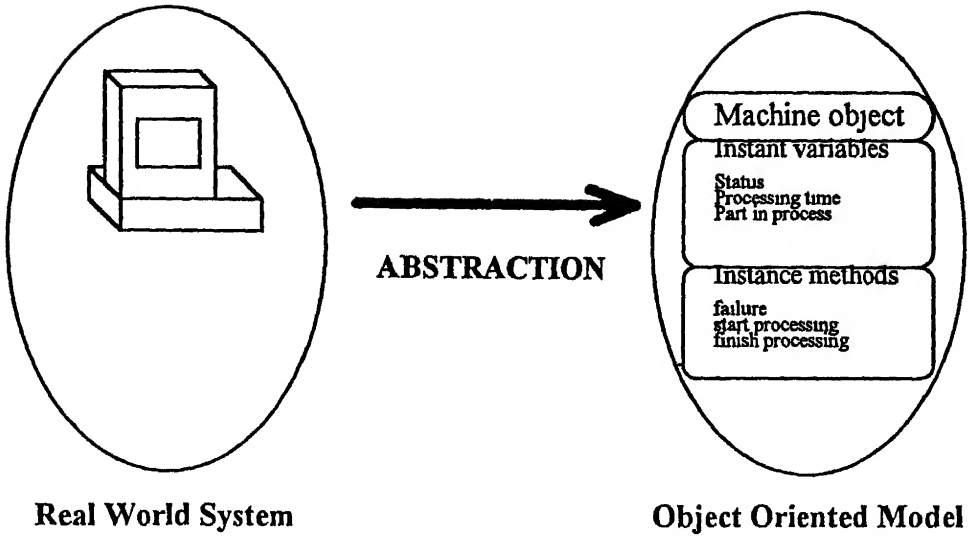


Figure 2.1: Modeling Abstraction

The data or information pertaining to this machine, such as status, part in process, processing time, jigs and fixtures, etc is stored in the instance variables of the machine object The machine failure, process starting or completion, and in general the real world machine behavior is captured by the instance methods of the machine object

In simulation, a real world interaction between the components of a manufacturing system becomes a dialog (via message passing) between the corresponding objects Workstations are made of basic modeling primitives an input queue, an output queue, a processor or machine, etc The first step in the process of developing an object oriented model of the entire manufacturing system is to identify primitives at the required level of abstraction The next step is to build the corresponding objects that represent these primitives Once the primitives are appropriately modeled, the process of bottom up model building begins By using the primitives as building blocks higher level coupled objects like workstations are created By appropriately combining the coupled objects, an entire plant (or plant object) can be fabricated

Modular infrastructure of object oriented models offers great advantages Modeling primitives along with the higher level objects can be stored in the object databases for later usage Plant reconfiguration or even new plant models can be developed by properly selecting the previously stored building block objects from the database A change in the real world system, such as introduction of a new machine in the current manufacturing system, can be quickly adapted by adding a corresponding machine object into the current model Thus, the object oriented modeling paradigm provides a highly reusable and quickly re configurable modeling environment

2.3.1 Advantages of Object Oriented Modeling of Manufacturing Systems

2 3 1 1 ***Reusability*** This advantage of OOM comes directly from the object oriented programming advantages such as encapsulation, inheritance, message passing, and late binding

2 3 1 2 ***Multiple levels of abstraction*** As the objective of a simulation study changes, a multi-level abstraction of the same system is inevitable It is possible to represent a plant in multiple levels of abstraction Every object knows its behavior and has the ability to respond to the external messages This capability of the object makes it possible for the user to run an overall simulation of the plant or a workcenter or a single workstation

2 3 1 3 ***Common framework*** Besides a change in objective, the user may wish to employ a variety of different analysis tools such as analytical models (queuing networks or petrinets) to obtain performance measures of manufacturing systems The idea here is to develop a common base representation to extract the appropriate analytical model via translation

2.3.2 Limitations of OOM

2 3 2 1 ***Language*** : Object oriented models are developed in OOP languages and hence, they automatically inherit the disadvantages of OOP languages slow processing and extensive computer resource requirements

2 3 2 2 ***Extensive library required*** Many authors including Adiga have fairly suggested that the real productivity improvement through OOP will only be realized when the user has an extensive library of modeling primitives from which to construct models

2.4 OBJECT ORIENTED PROGRAMMING LANGUAGES

As object oriented languages like C++, Smalltalk, Eiffel win acceptance, OOP is coming into the mainstream of computing Moreover object based languages such as Ada are moving into new object oriented languages such as Ada 9X one major consideration in deciding whether or not to commit to OOP and such languages as C++, Eiffel, Smalltalk, or Ada 9X is how programs written in earlier languages such as C, Pascal or FORTRAN Bjarne Stroustrop, who created C++, said he wanted something that run like greased lightening and allowed easy interfacing with rest of the world

In principle, object oriented programming concepts can be implemented in any programming language But it is done most conveniently in a language designed or extended to provide for objects, messages etc

C++, the exciting young language, blending the C language with support for object oriented programming, seems destined to be one of the most important programming languages of the 1990s Its C ancestry brings C++ the tradition of an efficient, compact, fast and portable language Its object oriented heritage brings C++ a fresh programming methodology designed to cope with the escalating complexity of modern programming tasks This dual heritage is both a blessing and a bane It makes the language very powerful, but it also means that there is much more to learn C++ joins together two separate programming traditions - the procedural language tradition, represented by C, and the object oriented language tradition, represented by the enhancements C++ adds to C

Table 2.1 : Contrasting Traditional and Object Oriented Paradigms for Simulation Modeling		
Factors	**Traditional Modeling Paradigm**	**Object Oriented Modeling Paradigm**
Model Construction :		
Software	Simulation languages based on procedural programming style	Simulation environments based on object-oriented programming style
Translation into code	Process is abstract	Process is natural and intuitive
Interface	Usually textual	Usually graphical with icons and dialog boxes
Level of detail	Usually not much detail due to programming complexity	At user's discretion, but requires detailed object library
Treatment of distinct system elements	Different element types are not distinctly modeled, Aggregation to reduce program complexity	Physical, informational and decision/control elements are modeled distinctly and independently
Effort/time/cost	Moderate costs of model development but a 'throw away' type	Initial cost of establishing detailed model is very high, but costs of subsequent reuse is relatively low
Model attributes :		
Purpose	Usually a unique model is created for a specific purpose	More general models possible for multiple purposes
Usage	Single usage, throw away models	Repeated usage and continuous refinement
Flexibility	Highly inflexible, changes almost always result in complete rewrite of programs	Highly flexible, due to the ability to modify fundamental building blocks, quick reconfiguration is possible
Accuracy	Useful for measuring relative differences in alternative configurations	With great degree of detail and realism, can also estimate absolute performance with great accuracy
System/Model relationship	Not connected via data links	Detailed model can be embedded in control structure of the firm, with linkages to databases, continuous model calibration and parameter updating

CHAPTER THREE

SYSTEM MODELING

3.1 SYSTEM DESCRIPTION

FMS considered for the present study is a very generalized one The major entities of the system are described below

3.1.1 *Work station*

Work stations are the processing stations where actual machining and assembling operations are carried out on parts The stations can accept only one job at a time Each of these stations is associated with an individual input and output buffer of limited capacity Input buffer stores parts which are to be processed by the workstation and output buffer stores parts which have been processed and are to be transported to some other workstation for further processing Workstation picks a part from the input buffer according to some dispatching rule and after processing it, drops it in the output buffer if it is not full If the output buffer is full, the workstation's status is blocked i e it can not do any further processing

3.1.2 *Loading Station :*

Parts coming for processing enter the system through the loading station which is associated with a loading buffer of relatively large capacity

3.1.3 *Unloading Station :*

Unloading station is the place from where finished parts are taken away as soon as they are unloaded at this station A part, after completing all its operations, reaches unloading station It is also associated with a buffer of large capacity

3.1.4 *Material Handling System :*

An automatic guided vehicle (AGV) based material handling system is considered for transportation of the parts from loading station to input buffers of

appropriate machines and from output buffers to input buffers or exit station Several AGVs may be employed to provide sufficient material handling capacity

3.1.5 ***Loader :***

Each workstation including the loading and the unloading station is equipped with one loader each The loader can do loading and unloading of jobs from buffer to the AGV or vice versa Loader can do either loading or unloading at a time For the purpose of modeling, the loader is assumed to be an integral part of the workstation

3.1.6 ***Parts :***

Parts are the raw material or semi finished parts on which processing is done by machines at the workstations to transform them into finished parts Each part belongs to a particular part family (type) and has an individual part number Though parts belonging to a particular part type follow the same sequence of operations, each part is differentiated by the individual processing time for each operation There is prespecified technological relationship among various part types i e a given part type may have one or more predecessors

3.2 ASSUMPTIONS

The following assumptions related to various entities are made in further characterization and modeling of the FMS

3.2.1 Parts :

(1) The number of part types which need to be processed are thought to be a apriori determined

(2) Part arrival process is assumed to be Poisson with known arrival rate

(3) The batch size is random between one and six for the parts which do not have any predecessors whereas for the parts having predecessors, batch size depends on the number of parts of predecessors lying in the unloading station

(4) Each part type possesses a sequence of operations to be performed An operation can be done on more than one machine thus providing flexibility in routing

(5) A part can revisit a machine but two consecutive operations on a machine are not allowed

(6) Operation times are assumed to be stochastic

(7) No pre-emption of operations is permitted, i e , once an operation on a part starts, it must finish before another operation can be performed on that part

(8) For an operation of a part, only one of the alternatives of each resource type is employed

3.2.2 ***Workstation :***

(1) Each workstation is self sufficient i e they are provided with tool magazines of sufficient capacity and the tools required by them are assumed to be loaded on them Tool changing is assumed to be automatic and requires negligible time or time which is a part of the processing time

(2) The set up times are assumed to be negligible or alternatively they may be considered as a part of the processing times

(3) Machine breakdowns are not considered

(4) Workstations accept only one job at a time for processing

3.2.2 ***Material Handling System :***

(1) There are sufficient number of AGVs in the system, so it takes fixed amount of time for transportation of parts which is independent of the source and destination

3.2.3 ***Supporting Equipment :***

(1) Pallets, fixtures are not explicitly considered assuming that they are available in sufficient quantity and thus does not affect the schedules

3.2.4 ***Loader :***

(1) Loading, and unloading times at the machines are assumed to be constant and are independent of the part types

3.3 SYSTEM CHARACTERIZATION

3.3.1 JOB ARRIVAL

As mentioned in the previous section, external jobs arrive at the system following a Poisson stream with a rate λ i e the inter arrival times of the jobs is exponentially distributed The different job types arrive at the system with different probabilities representing the production ratios Jobs arrive in batches There is a threshold No > 0 on the external job flow, such that whenever the total number of jobs at the loading station reaching No, external arrivals are turned away and lost In case of jobs which have predecessors, batch size is the maximum number which can be produced by the unused predecessors lying in the unloading station For parts with no predecessors, batch size is random between one and six Release times of the parts are the times when they enter the loading station Release times of all the parts of a particular batch are same

3.3.2 DUE DATE

Due dates are assigned to the parts when they enter into the system by the Total Work Content (TWK) method (Conway 1965) using the following definition

$$D = t + k P$$

where

D = due date of the job

t = time at which job enters the system

P = total average processing time of the job

k = parameter specified by the management (>=1)

3.3.3 FLEXIBILITIES

Two types of flexibilities have been explicitly modeled in the system

3.3.3.1 ***Routing Flexibility :***

Each part goes through an ordered sequence of operations, but it has a choice between more than one workstations to carry out each of its operations Alternatively, each workstation is capable of carrying more than one operation on different parts So each part within a part type can have different route thus providing routing flexibility Increasing the number of routes increases the routing flexibility of the system An increase in number of routes may also allow for more effective utilization of the machines thus affecting other flexibility types

3.3.3.2 ***Precedence Flexibility :***

The FMS takes into consideration the interrelationship of various part types while making the scheduling decisions This can be viewed as the problem of scheduling interdependent parts where a part can not be scheduled until all its predecessors are ready There is a prespecified technological precedence relationship among various part types Some of the parts, when ready, are assembled to form other parts All such parts, which have predecessors, are allowed to enter the loading station only when the unloading station contains all the predecessors and the batch size is determined by the maximum number of that part which can be produced by the predecessors in the unloading station For example, if part type 3 has two predecessors viz part type 1 and part type 2 If there are three parts of type 1 and two parts of type 2 in the unloading station, then the batch size for part type 3 would be two

3.3.4 CONTROL STRATEGIES

A number of control strategies have been incorporated in the model at various decision points These can grouped into two major sections as follows

(1) Rules for scheduling of parts for machine loading

(2) Strategy for machine selection for an operation

3.3.4.1 Part Selection :

The following are the types of decisions to be taken while scheduling the parts (i) Introduction of parts in the system, and (ii) Selection of job for machining These are briefly discussed in the following paragraph

3.3.4.1.1 ***Introduction of parts in the system :***

There is a threshold No on the number of parts that can be allowed into the system which depends on the capacities of input and output buffers associated with each workstation If M is the sum of the capacities of all the buffers associated with various workstations then the threshold No = M + Number of workstations - 3 Parts are allowed to enter the system from the loading station only when the total number of parts currently present in the system is less than No This is done to check congestion in the system

Of all the jobs that are present in the loading station, that job is selected to enter into the system which will go to a machine with least load i e the sum of processing times of all parts in its input buffer and the one which it is currently processing, should be least as compared to machines associated with other jobs in the loading station In case of a tie, job is selected randomly

3.3.4.1.2 ***Selection of jobs for machining :***

This concerns to the selection of job from the jobs which are waiting for processing in the input buffer of a particular workstation Five job dispatching strategies, which are very popular in the general job shop environment as well as FMS environment, are considered for comparison These rules are (i) FIFO, (ii) SIO, (iii) EDD, (iv) AT, and (v) WINQ

FIFO rule prioritizes jobs in a queue according to the ' First In First Out ' strategy i e the job which has spent the longest time in the queue has the right to be processed first

The shortest imminent operation (SIO) rule selects for processing that job for which the operation can be completed in the least time

EDD rule selects job on the basis of the due dates assigned to the jobs The job, which has the earliest due date, is given priority over other jobs for processing The principal advantage of due date based rule over processing time based rules is smaller variance of job lateness This advantage is specially manifest when due dates are established as some multiple of total processing time

AT rule bases its selection of job on the arrival times of the jobs The job which has entered into the FMS first, gets right to be processed first

All the dispatching rules which have been discussed above are job based i e they are based on the characteristics and attributes of the jobs They don't take into account the status of various machines present in the FMS But the last rule, WINQ, is a global dispatching rule which capitalize on the information about various machining centers in the FMS WINQ (Work in next queue) selects the job that will go on to its next operation at the machine with the least workload i e sum of processing times of parts in the input buffer is the least as compared to machines associated with other parts WINQ attempts to compete with SIO as it attempts to select jobs that can be processed rapidly through the next workstation

3.3.4.2 Machine Selection :

The following steps are involved in the selection of the machine for the job to visit

Step 1 For all the machines, which can process the selected job, calculate the sum of mean processing times of all the parts present in their input buffers and the parts currently being processed by them

Step 2 Select the machine which has the least sum of processing times i e the least workload

Step 3 In case of a tie in step 1, select the machine randomly

3.4 MEASURES OF PERFORMANCE

There are a wide variety of performance measures that are commonly used in manufacturing simulation studies, for example, throughput, makespan, WIP, equipment utilization, and the time jobs spend in queue The performance measures can be related to various entities of the system For the present work, performance measures related to workstations and jobs are considered

In an expensive system like FMS, high equipment utilization is of prime importance It is always desired that machines should be in use as long as possible Average machine utilization denote the percent of time the machines are busy on average From the job's point of view, job flow time i e the time a job spends in the shop is of concern Another measure is job waiting time which is the time spent by the job waiting in queues for processing It is the unproductive part of the flow time Low job flow and waiting times are characteristic of good scheduling in an FMS But industry considers meeting due dates to be of prime importance Job tardiness results are of practical significance to production managers Typically tardiness is represented by mean tardiness and number of jobs tardy Minimization of mean flow time does not minimize mean tardiness

Hence following measures of performance have been considered for the present work

Performance Measure	Symbol	Definition
Average utilization of machines	AMU	$\Sigma U_i / M$
Mean flow time	F	$\Sigma (P_i - r_i) / N$
Mean tardiness	T	$\Sigma \max (0 , L_i) / N$
Mean waiting time	W	$\Sigma W_i / N$

Notation

M = number of workstations in the FMS

N = number of jobs completed

U_i = the utilization of the i^{th} machine

r_j = the arrival time of the j^{th} job to the system

P_j = time at which job j is completed

L_j = lateness of the job

W_j = time spent by job j waiting in queues

3.5 SIMULATION

Simulation is commonly used by many researchers for the analysis of stochastic systems It is the conceptually simplest approach for modeling such systems Dynamic stochastic systems such as the modern manufacturing systems are too complex to be effectively modeled by analytical approaches Even if mathematical modeling of such complex systems is done, finding solution to the problem is practically impossible Further when the real time control and scheduling is required, which is there in the practical cases, the analytical modeling fails To develop an appropriate and cost effective FMS, one needs a design and planning tool that can consider the dynamic behaviors of the system Here simulation has been found to be very useful It allows one to develop an FMS design, and predict its performance Simulation helps us to evaluate the performance of a model of a system, which enables detailed representation of the characteristics of the studied system, to estimate the true behavior of the system Alternative proposed system designs can be compared via simulation to see which best meets a specified requirement

However, simulation is not without its drawbacks Each run of a stochastic simulation model produces only estimates of a model's true characteristics for a particular set of input parameters Simulation models are often expensive and time consuming to

develop Very often, distributions (e g normal or uniform) as input to simulation are very difficult to obtain But advantages of simulation outweigh disadvantages

Thus simulation provides comprehensive opportunity for the integration of design, planning and control functions, and is critical to successful implementation of FMS

3.5.1 SIMULATION LANGUAGES

3.5.1.1 *Special Purpose Simulation Packages :*

These packages are designed to be used by non programmers and contain good user interfaces but generally they are weak in control strategies Modification in the packages is extremely difficult Examples are MAP/I (Wortman & Miner 1986), SIMFACTORY (based on SIMSCRIPT , Klein 1986), XCELL (Jones and Maxwell 1986), FACTOR (Grant 1989), etc

3.5.1.2 *General Purpose Simulation Languages / Packages :*

High level languages such as GPSS (Schreiber 1974), SLAM (Pritsker & Pegden 1979), SIMSCRIPT (Caci), SIMAN (Pegden 1985) etc aim to make modeling a little easier The simulation languages mentioned above were implemented in FORTRAN / C as a collection of subroutines specified with a set of control statements This results in reduction in the amount of code one has to write to model a desired situation They help the users in simplifying the burden of handling repetitive tasks such as initialization, time advancements, etc Most simulation languages have the basic constructs needed to represent most common production facilities Inadequate user interfaces, weak development environments , inflexibility in representation systems are among some of the problems in using the current simulation languages

3.5.1.3 *General Purpose Languages :*

FORTRAN was, by far, the most popular language for implementing simulations in the 1970s because of the ability to handle numeric computations conveniently and efficiently In the recent years, PASCAL and C have emerged as competitors to

FORTRAN These newer general purpose languages provide some data structures that make the abstraction process a little easier An advantage in using the General purpose languages is the ability to analyze a virtually limitless number of problems

3.6 EVENT DESCRIPTION

The simulation model is constructed using a discrete event approach in which the state variables change instantaneously at separate points in time (In more mathematical terms, the system state can change only at countable number of points in time) Fixed increment time advance technique is used for advancing the simulation clock With this technique, the simulation clock is advanced in increments of exactly Δt time units for some appropriate choice of Δt After each update of the clock, a check is made to determine if any events should occur during the next interval of length Δt If one or more events are scheduled to occur during this interval, these events are considered to occur at the beginning of the interval and the system state (and statistical counters) are updated accordingly For the model considered , fixed interval Δt was taken to be unity The status of the whole system is checked at these discrete time units The simulation clock and the system status are updated each time The system flow chart is given in figure

The following events are considered for each value of simulation clock time (1) Arrival of job at the loading station, (2) Entry of job into the FMS, (3) Processing of jobs by the workstations, and (4) Updating of statistics

3.6.1 *Arrival of jobs at the loading station :*

For each iteration, it is checked whether current time is greater than or equal to the scheduled arrival time of next batch If it is, then length of the buffer of loading station is examined If it is less than or equal to six less than its maximum capacity, then parts can be allowed to enter the loading station The part type P, which is going to enter the system is determined by generating a random number between 0 and 1 and checking it against the

probabilities of various part types to enter the system If this part type has no predecessor, then the batch size B is random number between 1 and 6 If there are some predecessors, then the buffer of unloading station is checked for unused parts of each predecessor The batch size B is determined by that predecessor which has the minimum number of unused parts present in the buffer of the unloading station

B parts of part type P are then entered into the buffer of the loading station They are initialized and their release times and due dates are set according to current time and TWK rule respectively The time of next arrival is then scheduled

3.6.2 ***Entry of the job into the FMS :***

The current population of parts CNo in the FMS is calculated by adding the number of parts present in the input and output buffers of all the workstations and the parts currently being processed at various workstations If this current population is less than the threshold value No (Maximum number of parts allowed in the system which is three less than the sum of maximum capacities of input and output buffers of all the workstations and the number of workstations), then the part can be allowed to enter the FMS Of all the parts present in the loading station, that part is chosen which will go to the machine with the least workload (i e sum of mean processing times of parts present in its input buffer and the one currently being processed is least) and sent to the input buffer of the selected machine

3.6.3 ***Processing of jobs by workstations :***

At each clock time unit, status of all the workstations is checked If the machine is in IDLE state, then the Idle time associated with the machine is incremented by one and the status of its input buffer is checked If there are some parts present in the input buffer, then a part is selected from the input buffer according to the chosen dispatching strategy and loaded on to the machine The machine's status is set to BUSY and the actual processing time of the part is calculated by multiplying the mean processing time by a factor according to the probability distribution of processing time multipliers

If the status of the machine is BUSY, then the processing time of the part being processed is checked If it is more than one, then it is decremented by one unit and busy time associated with the part is incremented by one unit If the processing time of the part turns out to be zero, then the machine's status is assigned to be of IDLE state and the part is downloaded on the output buffer if it is not full If the output buffer is full then the idle time associated with the machine is incremented by one unit

Status of output buffer is checked at each time If there are parts present in the output buffer, then that part is selected, which will go to the next machine with the least workload, and sent to the input buffer of that machine provided it is not full If the part has already completed all its operations then the part is sent to the unloading station Its completion time is set to the current time

3.6.4 *Updating of Statistics :*

Though it is not a regular event, still it can be considered as an event When a job leaves the FMS and enters the unloading station then the counter and statistics for flow time and the lateness are updated Flow time is calculated by subtracting the Release time of the part from its Completion time If the part has completed all its operations beyond its due date, then the job is tardy and the tardiness of the job is calculated by subtracting the due date from the completion time of the part

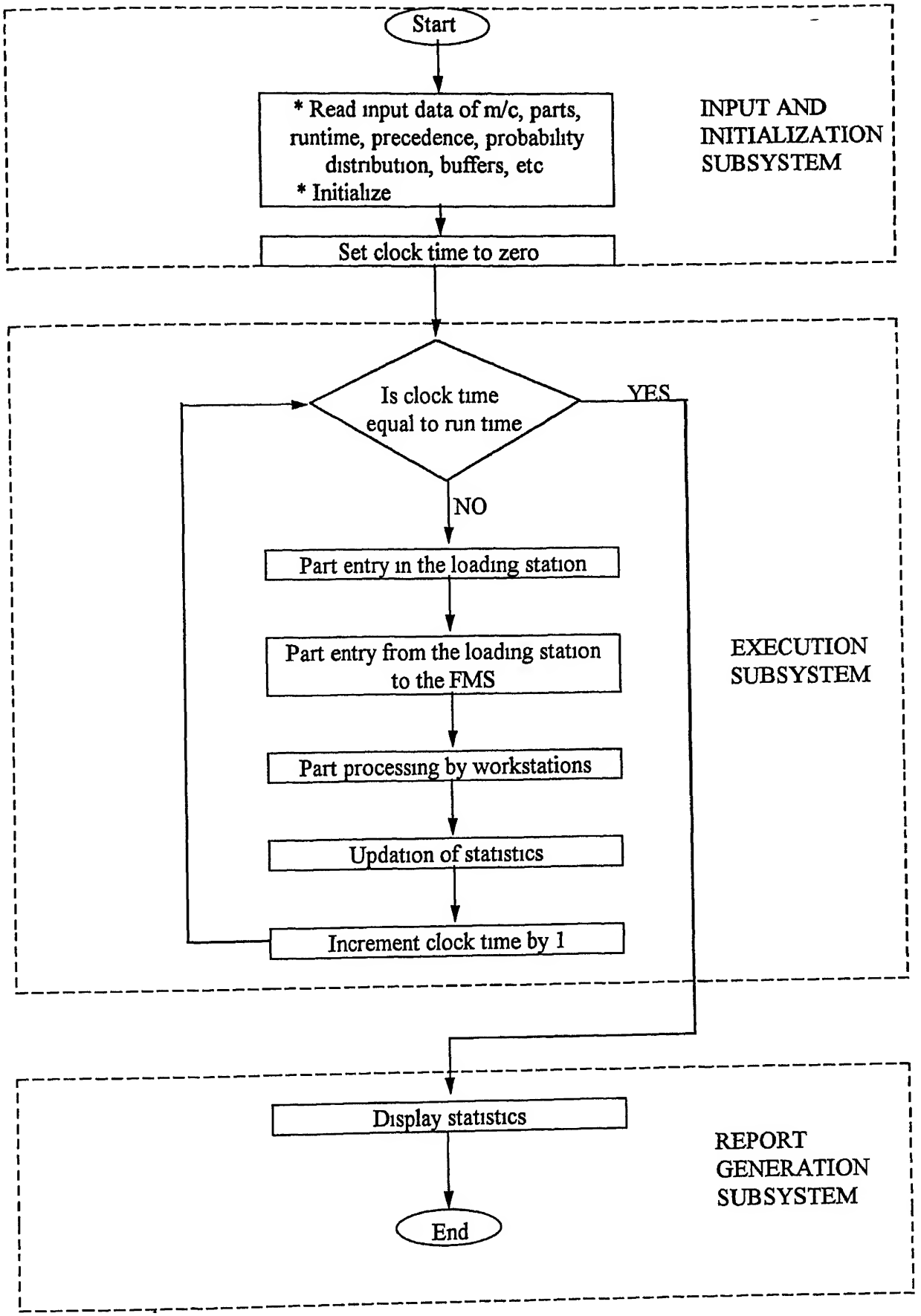


Figure 3.1 : Flow chart for the simulation model for FMS

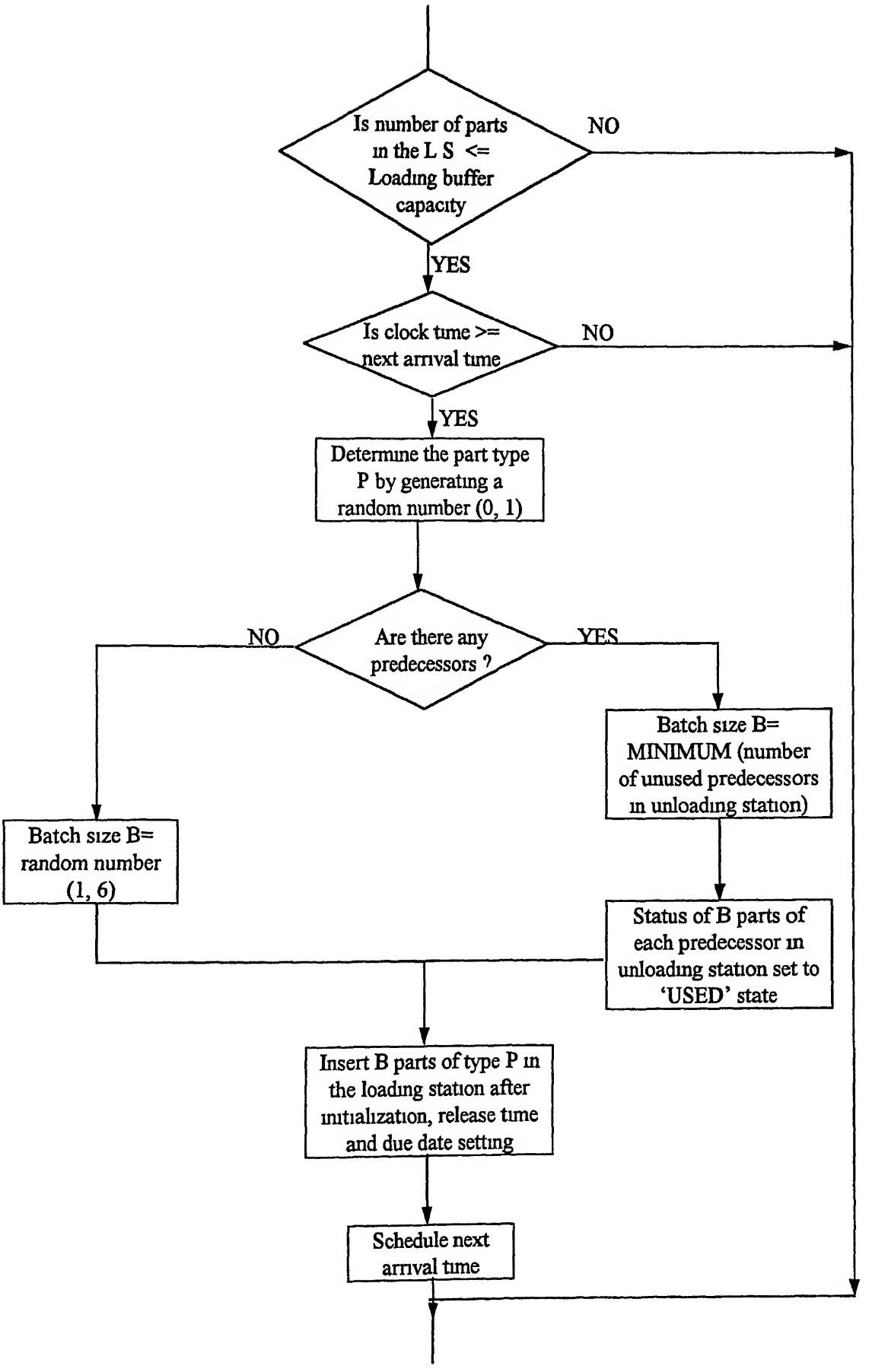


Figure 3.2 : Flow chart for part entry in the loading station

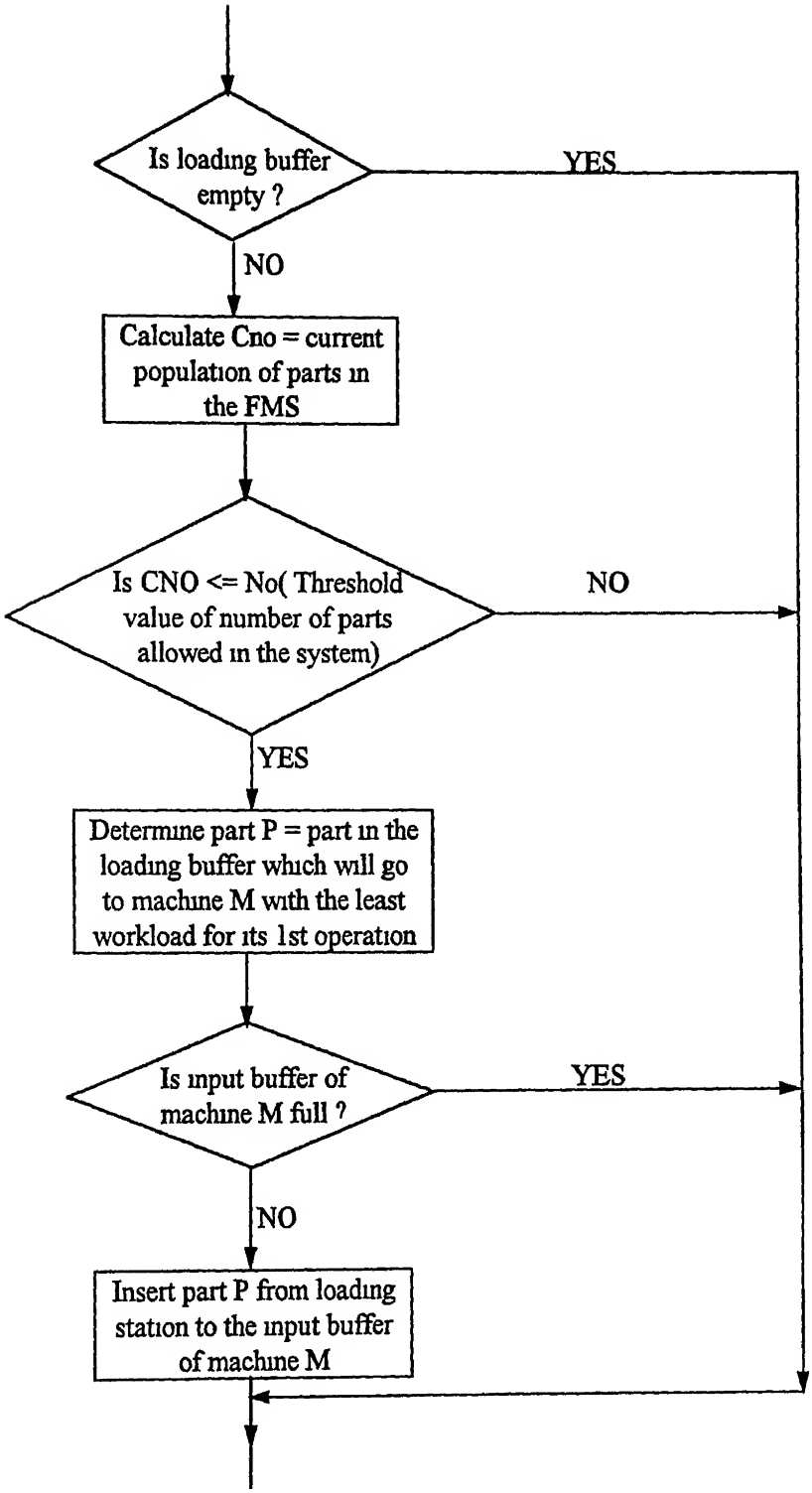


Figure 3.3 : Flow chart for part entry from the loading station into the FMS

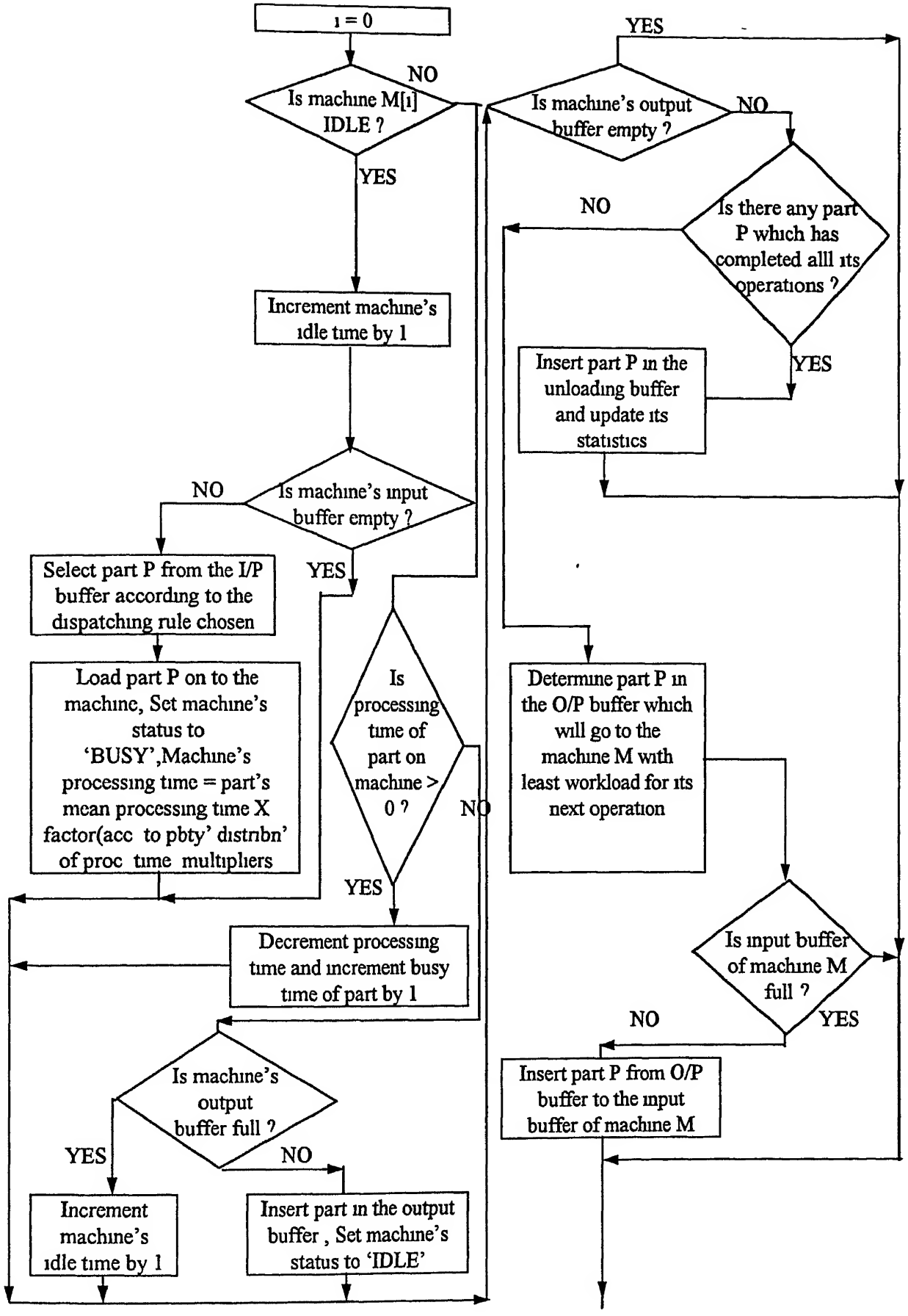


Figure 3.4 : Flow chart for processing of parts

CHAPTER FOUR

IMPLEMENTATION OF THE MODEL

The simulation model developed in the previous chapter has been implemented in C++ This chapter outlines the general features about the implementation

4.1 A WORD ABOUT THE IMPLEMENTATION MEDIA

There is nothing so difficult about writing a simulation program The essential components, such as a clock mechanism, random number generators , and data structures to represent transactions, resources, and queues, etc can all be implemented without serious problem Although it is not hard to write a simulation program in a general purpose language, it may be tedious To cope up with this lengthy process of building a simulation model, a number of special purpose simulation languages such as SIMULA, GPSS, SIMSCRIPT, etc have been developed and are widely used All these special purpose languages have passed through several versions None of them, however, can be construed as a modern language and each has a badly flawed design

Among the general purpose languages, though FORTRAN is mostly used in simulation, it does not have adequate data structures except array So keeping different attributes of an entity at one place is difficult On the other hand, Pascal and 'C' have highly sophosticated data structures and they also support dynamic allocation of memory Because of these reasons, data structures of linked list type can be created to handle the various events, and thus the memory requirement does not depend on the length of the simulation

But these languages do not support object oriented features of programming It has already been seen in chapter two that object oriented programming has several advantages over traditional programming For the present work, Borland C++ version 3 1 is used C++ is classified as a hybrid object oriented language, as opposed to a pure or orthodox object oriented one, because it is built on top of a more traditional procedural language 'C' So C++ consists of the whole of 'C' and much more which

qualifies it to become an object oriented language So all the advantages which are there in 'C' are also present in C++ also Other languages (such as Eiffel or Smalltalk) are pure languages, but that doesn't necessarily make them better What constitutes 'better' depends on how a language is to be used and on what task is to be solved C++ represents an excellent balance between power of expression, runtime speed, and code size Moreover , the Borland C++ Integrated Development Environment (IDE) automatically connects all the phases of the development cycle, speeding up the cycle dramatically Because C++ was designed as an improvement to and as an extension to 'C' , it is full of the traditional features of ANSI C

4.2 THE FRAMEWORK IN THE DEVELOPMENT OF FMS SIMULATION

The object oriented approach has the power to provide a more consistent approach for software development It builds on the strength of the traditional techniques, and emphasizes data abstraction, encapsulation, information hiding, inheritance, and reuse In the development of FMS simulation using object oriented programming method, a framework is built It defines two phases for the software development the design phase of simulation and the realization phase of the simulation

4.2.1 THE DESIGN PHASE OF SIMULATION

The design phase considers the system as a solution to a problem in its environment During this phase, the fundamental objects within the FMS are identified, and their object classes are defined The attributes of each class are also identified

As FMS is a discrete event system, the objects in the system can be easily recognized, such as machine, part, buffer, and so on In order to make the system run, sometimes additional objects are required For example, object 'clock' is needed in the simulation

Having got the 'objects' in the system, the relationships between these objects should be analyzed in detail The goal is to capture the object behavior from an

external viewpoint and determine what member functions (methods) they need As soon as the member functions of the objects are determined, both the attributes of the object and other necessary data, can also be determined Then by using object oriented language, the object's classes in the software are implemented

A class can have three types of members viz private, public, and protected The private data can be accessed only by the class members, private or public A program can access the private members of an object only by using the public member functions Thus, the public member functions act as go-betweens between the program and an object's private members , they provide the interface between object and the program This insulation of the data from direct access by a program is called *'data hiding'* The 'protected' keyword is like 'private' in that a program can access class members in a protected section only by using public class members The difference between protected and private comes into play only with classes derived from the base class Members of a derived class can access protected members of the base class, but they cannot access the private members of the base class

The program design can be visualized in terms of a *'client server'* model In this conceptualization, the client is a program that uses the class The class definition, including the class methods, constitute the server, which is a resource available to the programs that need it The client uses the server through the publicly defined interface only This mans that the client's only responsibility, and, by extension, the client's programmer's only responsibility , is to know that interface The server's responsibility is to see that the server reliably and accurately performs according to the interface Any changes the server designer makes to the class design should be to details of implementation, not to the interface This allows the programmers to improve the client and the server independently of each other without changes in the server having unforeseen repurcussions in the client's behavior The main task in the design phase of the simulation is to implement the server

There can be two types of objects physical and information A *'physical object'* is an object with a tangible correspondent in the real world system There is a

natural one-to-one correspondence between physical classes in a manufacturing system and software classes that represent them From the description of a conceptual framework used for the simulation of a manufacturing system, we can identify the software classes required by the simulation model of the manufacturing system An *'information object'* is an object which may or may not have a tangible correspondent in the real world system An object can be classified as an information object if the primary focus of the modeler's interest in the object is its information content The classes which were identified for the FMS are described below

4.2.1.1 PHYSICAL CLASSES

Three classes are identified which have one-to-one correspondence with the real world entities These are

4.2.1.1.1 Part :

Part is semi finished or raw material which has a fixed number of operations to be performed upon by machines Their private data include

(i) **int type**

type (family) of the part to which it belongs

(ii) **int number**

individual number of the part within that type

(iii) **int Rj**

Release time of the part

(iv) **int Dj**

Due date of the part

(v) **int Cj**

Completion time of the part

(vi) **int Tj**

Tardiness of the part

(vii) **int Fj**

Flow time of the part

(viii) **int use**

variable to check whether the part has been used for production of its successors or not

(ix) **int busy_time**

variable to keep track of the time for which the part is busy i e being operated upon by some machine

(x) **int tot_op**

total number of oerations for the part to complete

(xi) **int no_op_done**

number of operations completed by the part

The public member functions, which provide the public interface for the class are as follows

(i) **part()**

constructor of the class part for constructing the new class and assigning values to its data members Constructor of part assigns value 0 to no_op_done, busy_time, tot_op, use, Rj, Cj, Dj, Fj, Tj and 999 to 'type' and 'number'

(ii) **~part()**

destructor of the class part for the epiry of the object at the time of the program termination

(iii) **init(int t, int n)**

to assign the values 't' and 'n' to the data members 'type' and 'number' of the part respectively

(iv) **int type_r()**

returns the value of 'type' of the part

(v) **int Rj_enter(int k)**

assigns the value 'k' to the 'Rj' member of the part

(vi) **int Cj_enter(int k)**

assigns the value 'k' to the 'Cj' member of the part

(vii) **due_date(float f, int t, int D)**

assigns the value ' t + (D x f) ' to the due date member 'Dj' of the part according to the TWK rule

(viii) **int waiting_time()**

calculates the total time spent by the part waiting in buffers and returns it

(ix) **t_ops(int e)**

assigns the value 'e' to the member 'tot_op' of the part

(x) **int ret_Tj()**

returns the tardiness of the part

(xi) **int ret_Fj()**

returns the flow time of the part

(xii) **int ret_Dj()**

returns the due date of the part

(xiii) **int ret_Rj()**

returns the release time of the part

(xiv) **int is_ops_comp()**

returns the value 1 if 'no_op_done' is equal to 'tot_op' i e the part has completed all its operations and 0 otherwise

(xv) **op_inc()**

increments the value of 'no_op_done' by one

(xvi) **int is_part-used()**

returns the value 1 if the part has been used for the production of its successors and 0 otherwise

(xvii) **use_set()**

sets the part to 'USED' state i e assigns value 1 to the member 'use' of the part

(xviii) **btime_assign(int d)**

increments the value of member 'busy_time' by 'd'

(xix) **int eq_check(part & q)**

returns value 1 if the part is equal to the part q i e their 'type' and 'number' are same, and 0 otherwise

(xx) **update()**

calculates the values of flow time 'Fj' and tardiness 'Tj' of the part

4.2.1.1.2 Buffer :

Buffer represents the storage place for the parts It is implemented using the data structure of linked list of nodes Its private data members include

(i) **struct node**

:ts members are a part P and a pointer to the next node

(ii) **node *head**

pointer to the starting node of the list

(iii) **node *pos**

pointer to a particular node of the list

(iv) **int max_lgth**

denotes the maximum capacity of the buffer i e the maximum number of nodes it can hold

(v) **int curr_lgth**

denotes the current length of the buffer i e the current number of nodes in the buffer

The public member functions of the class buffer are as follows

(i) **Buffer()**

constructor of the class which assigns NULL value 'head' and 0 to 'curr_lgth'

(ii) **~Buffer()**

destructor of the class Buffer

(iii) **max_lgth_set(int p)**

sets the value of 'max_lgth' to 'p'

(iv) **insert(part p)**

inserts a new part 'p' in the buffer If the buffer is empty , it creates a new node, assigns its part to value 'p', increases curr_lgth by one, otherwise it inserts a new node at the end of the list

(v) **part remove(int n)**

removes the nth node, starting from the first node, of the list It returns the part of that node and decrements the value of 'curr_lgth' by one

(vi) **int ret_pt_type(int k)**

returns the 'type' of the kth part in the list starting from the first node

(vii) **int ret_cu_lgth()**

returns the current length of the list i e the present number of parts in the buffer

(viii) **int ret_pt_op(int k)**

returns the value of the number of operations completed by yhe kth part in the list starting from the first node

(ix) **int is_buf_empty()**

returns value 1 if the buffer is empty, and 0 otherwise

(x) **int is_buf_full()**

returns value 1 if the buffer is full i e its current length is equal to the maximum length, and 0 otherwise

(xi) **int EDD()**

returns the number of the part in the list starting from the first node, which has the earliest due date

(xii) **int AT()**

returns the number of the part in the list starting from the first node, which has the least value of release time 'Rj'

(xiii) **int check(int *pl)**

'pl' is an array consisting of the part types which are predecessors to a particular part type This function checks if all the predecessors are present in the buffer and returns value 1 if they are present , and 0 otherwise

(xiv) **int no_enter(int *pl)**

returns the minimum among the numbers of predecessors (given in array 'pl') present in the buffer

4.2.1.1.3 Machine :

The physical object of 'workstation' has been implemented in the class 'machine' The private data members of the class machine are as follows

(i) **int type**

represents the type of the machine

(ii) **int status**

machine can be in one of the two states viz 'BUSY' when it is processing parts, and 'IDLE' when it is waiting for parts This data member denotes the status of the machine

(iii) **int total_delay**

keeps track of the time for which the machine is IDLE

(iv) **part p**

denotes the part which is currently being processed by the machine

(v) **int proc_time**

represents the time left for the machine to complete the processing of the part p

(vi) **Buffer I_buffer**

denotes the input buffer for the workstation

(vii) **Buffer O_buffer**

denotes the output buffer for the workstation

The public member functions which define the public interface for the software class machine are defined below

(i) **machine()**

constructor for the class which assigns value 0 to the data members type, total_delay, proc_time, and assigns the status of the machine to the IDLE state

(ii) **~machine()**

destructor for the class machine

(iii) **type_set(int i)**

assigns the value 'i' to the data member 'type' of the the machine

(iv) **stat_assign(int k)**

assigns the value 'k' to the 'status' of the machine

(v) **int ret_delay()**

returns the value of the delay time i e time interval for which machine has been idle

(vi) **int ret_ptype()**

returns the type of the part currently being processed by the machine

(vii) **int is_mc_idle()**

returns 1 if the machine is in IDLE state, and 0 otherwise

(viii) **Icap_set(int c)**

sets the maximum capacity of the input buffer of the workstation to value 'c'

(ix) **Icap_set(int c)**

sets the maximum capacity of the input buffer of the workstation to value 'c'

(x) **int is_Ibuff_empty()**

checks if the input buffer is empty and returns value 1 if it is, and 0 otherwise

(xi) **int is_Obuff_empty()**

checks if the output buffer is empty and returns value 1 if it is, and 0 otherwise

(xii) **int is_Ibuff_full()**

checks if the input buffer is full and returns value 1 if it is, and 0 otherwise

(xiii) **int is_Obuff_full()**

checks if the output buffer is full and returns value 1 if it is, and 0 otherwise

(xiv) **pt_assign1(int k)**

assigns value 'k' to the data member 'proc_time' of machine which is the processing time left for the part currently being processed

(xv) **pt_assign2(int k)**

decrements the value of 'proc_time' by 'k'

(xvi) **delay_inc(int k)**

increments the delay time of the machine by value 'k'

(xvii) **int process_time()**

returns value 0 if the value of 'proc_time' is 0 , and 1 otherwise

(xviii) **int ret_proc()**

returns the value of the processing time left for which the machine is to operate on the part

(xix) **show()**

displays the values of various data members associated with the machine

4.2.1.2 INFORMATION CLASSES

A number of 'Information classes' have been designed to record the values of input relating to precedence, processing times, operations, routes, etc A brief description of such classes follows

4.2.1.2.1 List :

incorporates 'linked list' of nodes to record the input relating to operations of various part types, alternative machines on which operations can be performed and associated mean processing times Members of class List belong to 'protected' class As class 'List' intends to be base class for other classes, its private members have been declared protected The protected members of List are as follows

(i) **Struct Node**

defines the node of the linked list Its members are

(a) **int pt_type** denotes the type of the part for the node

(b) **int opn_no** denotes the operation number

(c) **int mc_type** denotes the machine type on which a particular operation can be performed

(d) **int proc_time** denotes the mean processing time for an operation on a machine type for a particular part type

(e) **Node *next** pointer to a node

(f) **Node *point** pointer to a node

(g) Constructor of the node assigns value 999 to pt_type,opn_no, and mc_type, 0 to proc_time, and NULL to 'next' and 'point'

(ii) **Node *Header**

pointer to the first node of the list

(iii) **Node *Crt_pos**

pointer to a node

(iv) **int prt**

denotes the total number of part types available for the FMS

(v) **int *noop**

represents an array of integers which stores the total number of operations for various parts

The public interface for the class List is as follows

(i) **List()**

constructor of the class

(ii) **~List()**

detructor of the class

(iii) **Data_entry()**

This member function os responsible for the creation of the llnked list Nodes can be classified of three types (1) nodes relating to part types, (2) nodes

relating to operation numbers, and (3) nodes relating to alternate machines for various operations and their corresponding mean processing times The list is created according to Fig 4 1

For each part type, type 1 nodes are created and linked together via 'next' For each part type, a separate list of type 2 nodes is created , the number of nodes being the total number of operations for that part type, and the nodes are linked together via 'next' This list is linked to the node corresponding to part type via 'point' In a similar way, nodes for alternate machines and corresponding processing times are created and linked via 'next' for a particular operation of a part type and joined to the node of that operation via 'point'

(iv) **int proc(int m, int p, int o)**

returns the value of mean processing time for the oth operation of part type p on the mth machine

(v) **int no_pt_type()**

returns the number of part types alloted for the FMS

(vi) **int no_op(int pt)**

returns the total number of operations to be performed by part typr 'pt'

(vii) **float part_avg_time(int pt)**

returns the average of mean processing times of a particular part type 'pt' for all operations over all alternate machines

(viii) **int *Mc_to_go(int pt, int o)**

returns the list of machine types to which part type 'pt' can go for its oth operation

4.2.1.2.2 p_list

The class p_list is a derived class inherited from the class List It is used for the entry of information regarding the predecessors of various part types As it is a derived class of the class List, all protected members of the class List become its protected members In addition, there are two more public members

(i) **P_data_entry(int P)**

For each part type, this function enters the information regarding its predecessors It uses data member 'noop' to store number of predecessors for each part type The manner in which linked list is created, is very much like that of class List Nodes are created for each part type and linked to each other For each node, which represents a part type, nodes are created which equal in number to the number of predecessors, and are linked to each other by a different link Information regarding predecessors are then fed in each node Fig 4 2 represents the structure of the list

(ii) **int *Nu_pred(int P)**

returns the list of predecessors for a particular part type 'P'

4.2.1.2.3 Pb_dis

It is the class responsible for the input data regarding the discrete probability distribution of processing time multipliers Its mode of implementation is also based on the linked list data structure Its private members are

(i) **struct Node**

members of Node are

(a) **int pt_type** denotes the part type for which the probability distribution is entered

(b) **float factor** denotes the multiplying factor

(c) **float pbty** denotes the discrete probability for the factor

(d) **Node *next** pointer to a node

(e) **Node *point** pointer to a node

(ii) **Node *start**

pointer to the first node of the list

The public interface of the class is as follows

(i) **Pb_dis()**

constructor of the class which assigns NULL value to 'start'

(ii) **~Pb_dis()**

destructor of the class

(iii) **Entry(int P)**

Nodes are classified into two types (1) nodes related to part types, and (2) nodes related probability distribution of processing time multipliers First, nodes of type 1 corresponding to each part type are created and linked together via 'next' Then for each node, nodes of type 2 are created, data entered for factor and probability, and linked together via 'point' till the sum of probabilities equals 1 The structure of the list is represented by Fig 4 3 After data entry in such a way, probability of each node is recalculated by adding the probability of the previous node to itself Thus, cumulative probability is entered in each node

(iv) **float ret_factor(int P)**

returns the multiplying factor to be multiped by the mean processing time to determine the actual processing time for a part type A random number between 0 and 1 is generated and the value of 'factor' of that node is returned for which the cumulative probability turns out to be more than the random number

4.2.1.4 clock :

This class is responsible for the timing mechanism during the simulation run It represents the 'simulation clock' The private members the class clock are as follows

(i) **int time**

to keep track of the current value of the simulated time

(ii) **int time_last_event**

represents the time at which the last event took place

(iii) **int count**

represents the number of parts at the unloading station during a particular time interval

The public member functions of the class are

(i) **clock()**

constructor for the class which initializes 'time' and 'time_last_event' to value 0

(ii) **~clock()**

destructor of the class

(iii) **int time_adv()**

returns the time elapsed since the last event took place

(iv) **count_inc()**

increments the value of count by one

(v) **time_inc()**

increments the value of time by one

(vi) **nt_assign()**

assigns the current value of time to time_last_event

(vii) **int count_check(int k)**

returns 1 if value of count is greater than or equal to k, and 0 otherwise

(viii) **int time_show()**

returns the current value of time

(ix) **int ret_count()**

returns the current value of count

(x) **float util(int k)**

returns the utilization of the machines by dividing their busy time by current 'time'

(xi) **show()**

displays the values of count and total processing time

4.2.1.3 FRIEND FUNCTIONS

Apart from the member functions associated with various classes, there are also a few friend functions A *'friend function'* is a non-member function that is granted access to a class's private members One of the most common reasons for using a

friend function is when a function is needed to access more than one classes Friend function belongs to none of the classes Friend function operates upon the objects passed to it as arguments The friend functions designed for the FMS simulation are as follows

4.2.1.3.1 int pr_tm_sum(machine & m, List & l)

This function is a friend to classes 'machine' and 'List' It takes in as arguments, objects of classes 'machine' and 'List', and returns the sum of the mean processing times of parts lying in the input buffer of the machine for corresponding operations on that machine and processing time left for the part currently being processed

4.2.1.3.2 fr(machine & m, List & l, int L, int t)

This function is also a friend to classes 'machine' and 'List' Its arguments are objects of classes 'machine', 'List', code for selected dispatching rule L This function loads a part from the input buffer of the machine according to a selected dispatching rule (1 FIFO , 2 SIO , 3 EDD , 4 AT , 5 WINQ) and assigns status of machine to the BUSY state It also updates the processing time and number of operations of the loaded part

4.2.2 THE REALIZATION PHASE OF THE SIMULATION

Through the design phase of the simulation, the objects' classes needed in the simulation are developed But they are isolated ones In the realization phase, objects of various classes are created and used in the main() function to simulate FMS Just as the FMS is controlled by the monitor, whose decision is made in the light of the schedule plan, the FMS simulation software derives the simulation according to the designer's schedule plan Function main() is needed to represent the FMS which acts as a 'client' using the 'server' classes to create objects and simulate FMS

4.2.2.1 SOFTWARE CONFIGURATION

The schematic diagram of the configuration is shown in Fig 4 4 The configuration of the simulator is composed of three subsystems

(1) Data Input and Initialization Subsystem,

(2) Execution Subsystem, and

(3) Report Generation Subsystem

These subsystems are described in the following subsections

4.2.2.1.1 Input and Initialization Subsystem

The data inputs are organized into three groups (i) inputs related to workstations, (ii) inputs related to part types, and (iii) control options

In the group of inputs related to workstations, objects of class machine are created and initialized for the number of workstations in the FMS Capacities of the input and output buffers of various workstations and loading , unloading stations are also initialized

In the group of inputs related to part types, an object of class 'List' is created and information regarding operations, alternate machines and corresponding mean processing times is entered through its member function The probabilities at which various part types will arrive, are also entered Objects of classes 'Pb_dis' and 'p_list' are created and information regarding probability distribution of processing time multipliers and predecessors of various part types are entered For the control options, information regarding the selected dispatching rule, mean interarrival time of the parts, runtime of the simulation and due date factor is entered An object of class 'clock' is also created and initialized

4.2.2.1.2 Execution Subsystem

This is the main module of the simulation system which is responsible for the running of the simulation Various events such as arrival of jobs at the loading station, entry of jobs into the system, processing of jobs by the workstations are implemented as described in the previous chapter using the friend functions and the member functions of various objects created

4.2.2.1.3 Report Generation Subsystem

After the simulation of the FMS is over, various measures of performance related to workstations and parts which include utilization of various workstations, their average utilization, mean part flow time, mean part tardiness, mean part waiting time and number of tardy jobs are calculated and displayed

4.3 INITIAL DATA DELETION

For the comparison of results of simulation, the methodology most often employed is to simulate the operation of the shop for a specific length of time and to collect the statistics on jobs that are completed Common statistics are mean flow time, mean tardiness, etc The initial conditions for a terminating simulation generally affect the desired measures of performance Output data for various measures of performance should be analyzed only when they are in steady state Care must be taken in choosing appropriate initial conditions Since the simulated FMS has no parts at time 0, first few days will show highly transient behavior with respect to measures of performance The technique used for dealing with this problem is ' *warming up the model* ' or ' *initial data deletion* ' in which some number of observations is deleted from the beginning of a run and the remaining observations are used to estimate the measures of performance For the present work, data generated during the first five days (8 hours per day) is discarded and status of the FMS after five days is treated as initial conditions Then the simulation is run for a specified time and various measures of performance are estimated So the 'warm up period' of the simulation is taken to be five days

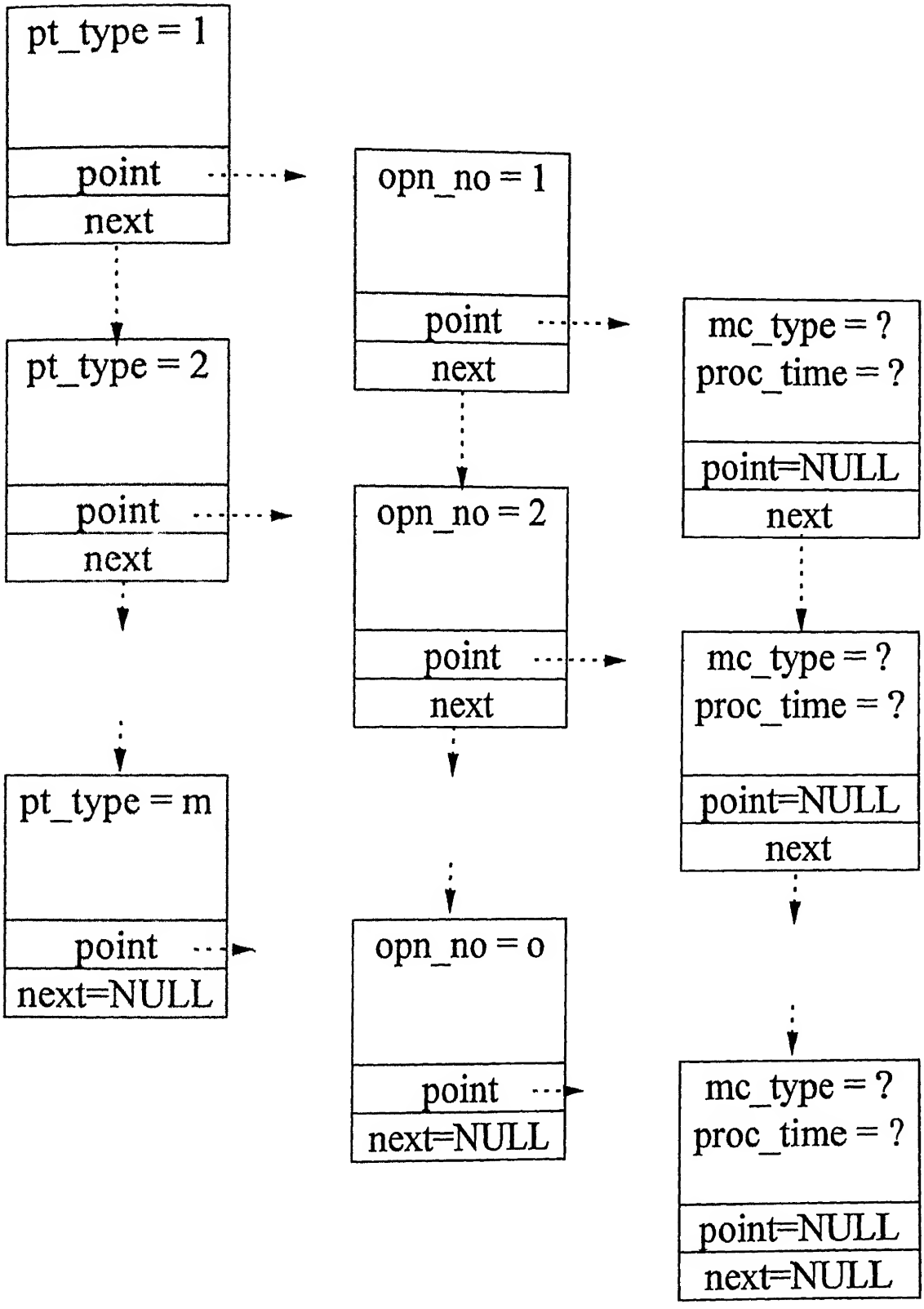


Figure 4.1 : Linked list representation of input relating to operations, related machines, processing times for various part types

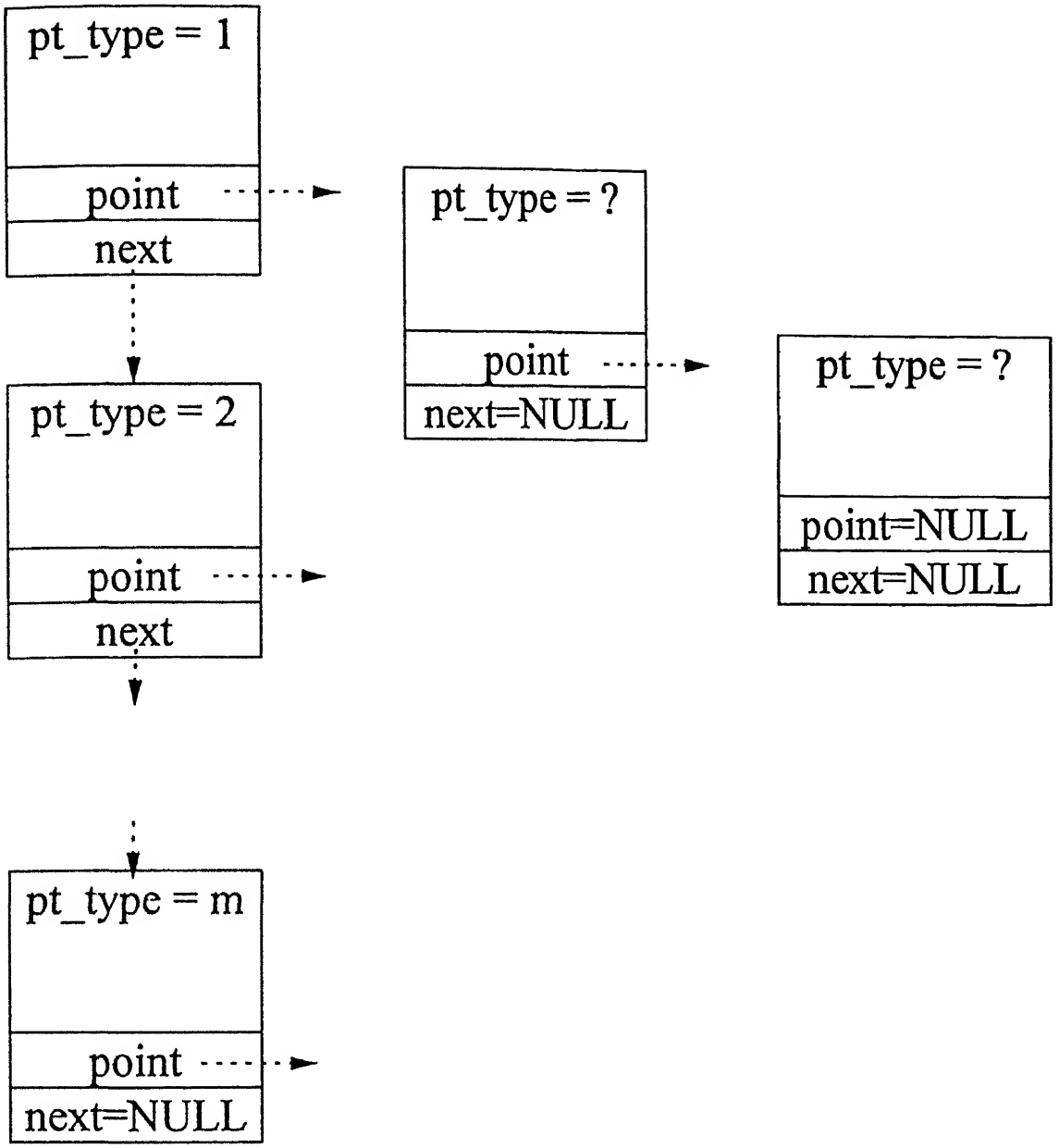


Figure 4.2 : Linked list representation of input relating to predecessors of various part types

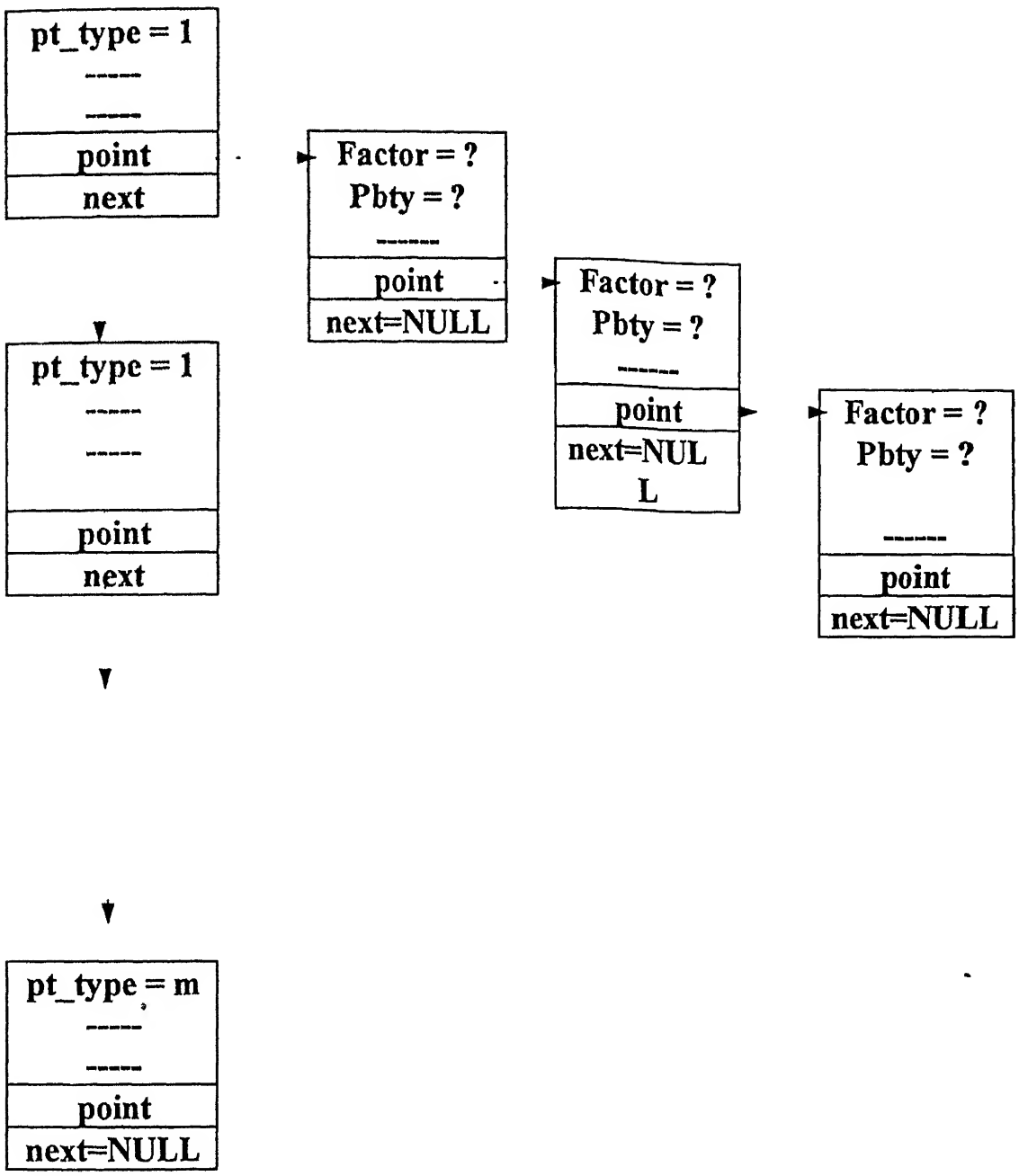


Figure 4.3 : Linked list representation of input relating to probability distribution of processing time nultipliers

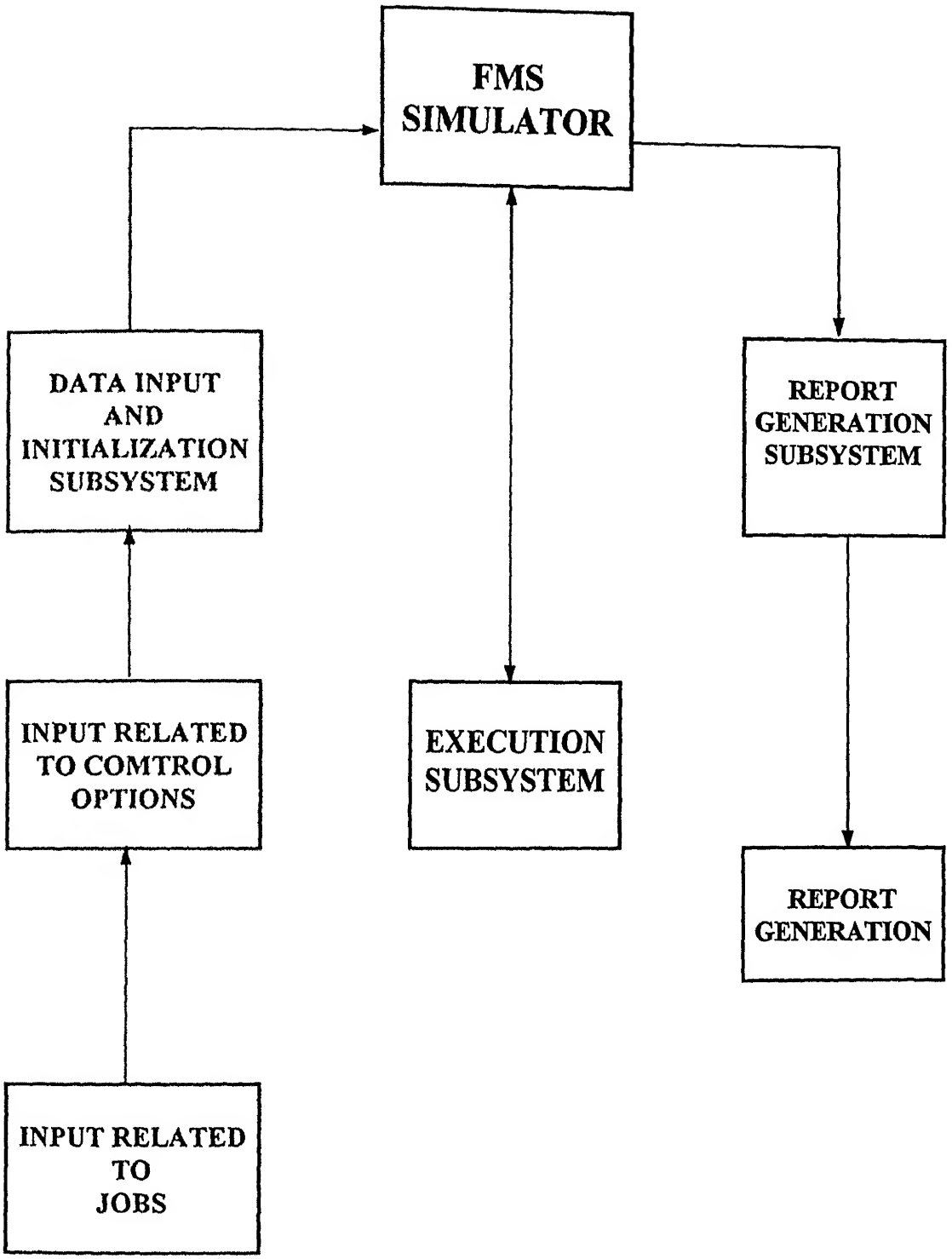


Figure 4.4 : Software configuration of the simulator

CHAPTER FIVE

EXPERIMENTATION : RESULTS AND DISCUSSION

In order to compare various dispatching rules, test runs have been carried out using the simulation model discussed in Chapter III with implementation details mentioned in Chapter IV Section 5 1 gives the details of the system configuration used for experimentation

5.1 SYSTEM DESIGN AND SPECIFICATIONS

The following system configuration including the details about FMS machines, job types, control strategy, flexibility and precedence considerations, and measures of performance is considered

5.1.1 Size of FMS

As mentioned in their publication of '*Collection of European and American FMS* (1981)', the Japanese Production Technology Investigation Society collected data on 79 existing FMSs from USA and Europe Table 5 1 shows the distribution of the number of FMSs against the number of machines in the system Japanese FMSs, which are not included in this table are generally smaller than those in the USA and Europe although Japan leads in terms of the number of FMSs (Barash 1978) With the above in view, four machines are included in the FMS Each machine has input and output buffer of size 8 each All the machines are available for the whole shift

5.1.2 Job types

The following characteristics related to jobs are included in the test runs

- Jobs arrive at the loading station
- Arrival process is Poisson An independent and identically distributed exponential inter-arrival time is selected with arbitrarily chosen mean of 75 minutes

- There are four types of jobs assumed to arrive at the system with probabilities of 0 2, 0 3, 0 3, and 0 2 respectively
- Enough number of AGVs are assumed to be running in the system So it takes a fixed amount of time, which is one minute, for a job to reach from a machine to another machine
- Buffer at the loading station has a maximum capacity of 20 and buffer at the unloading station has infinite capacity to accommodate all the parts that are produced during the simulation run time
- The maximum number of jobs which can be allowed in the system is taken to be 65 for the purpose of this analysis It is three less than the maximum possible number of parts in the system
- There can be technological relationships among jobs i e a part can have several predecessors A part can enter the system only when all its predecessors have completed all their operations
- Revisit on a machine for consecutive operations is not permitted
- Operations of a job type can be performed on more than one machine
- Only the mean processing times are provided Exact processing times are calculated at runtime by multiplying the mean processing time by a factor according to probability distribution of processing time multipliers defined for each part type
- Batch size is random between one and six for jobs having no predecessors whereas for jobs having predecessors, batch size depends on the number of unused predecessors present at the unloading station It is the minimum among the numbers of unused predecessors present at the unloading station
- Due dates are assigned in proportion of the total work content For the experimentation purpose, the due dates are taken as two times the total average processing time

5.1.3 Control options

The control options considered here for the experimentation are for selecting a job among the candidate jobs waiting for processing at the input queue of a machine They are (i) FIFO, (ii) WINQ, (iii) EDD, (iv) AT, and (v) SIO

5.1.4 Flexibility

The outputs are obtained for various levels of flexibility Flexibilities for the present analysis are grouped into three levels

1 *no flexibility* (an operation of a job can be performed on only one machine)
2 *medium flexibility* (on an average, an operation can be performed on 2 or 3 machines)
3 *high flexibility* (each operation can be done on any of the four machines available)

5.1.5 Precedence

For each level of flexibility, outputs are obtained for three levels of precedence They are

1 *no precedence* (all the four part types have no predecessors)
2 *medium precedence* (two of the four part types have one predecessor each Part type '1' is the predecessor for part type '2' and part type '3' for part type '4')
3 *high precedence* (a chain type precedence exists among part types Part type '1' is the predecessor for part type '2', part type '2' for part type '3', and part type '3' for part type '4')

5.1.6 Measures of performance

The following measures of performance are used to compare the dispatching rules

1 Average machine utilization (AMU)
2 Mean flow time for parts (MFT)
3 Mean tardiness for parts (MT)

4 Mean waiting time for parts (MWT)

5.2 EXPERIMENTATION

The experiments are designed in such a way that for a given set of input data, flexibility is varied from no flexibility level to high flexibility level and for each flexibility level, degree of precedence is varied from no precedence to high precedence Experiments are numbered in such a way that first digit denotes the level of flexibility (0 zero, 1 medium , 2 high) and second digit denotes the level of precedence (0 zero , 1 medium , 2 high) Input data which is common to all the experiments is as follows

(i) Number of workstations - 4

(ii) Number of part types = 4

(iii) Mean inter arrival time = 75 minutes

(iv) Capacity of the buffer at the loading station = 20

(v) Capacity of input and output buffers at workstations = 8

(vi) Probabilities of arrival of various part types

Part type '1' - 0 2

Part type '2' 0 3

Part type '3' - 0 3

Part type '4' = 0 2

(vii) Probability distribution of processing time multipliers for various part types is given in Table 5 2

(viii) Due date factor = 2

(ix) Job processing data for experiments with varying flexibility are given in Tables 5 3, to 5 5

5.3 RESULTS

The results of various experiments viz Ex00, Ex01, Ex02, Ex10, Ex11, Ex12, Ex20, Ex21, and Ex22 are given in Tables 5 6, to 5 14 Effects of varying precedence levels for various performance measures at various levels of flexibility are graphically

illustrated in Figures 5 1, to 5 12 and effects of varying flexibility levels for various performance measures at various levels of precedence are graphically illustrated in Figures 5 13, to 5 24

5.4 OBSERVATIONS

From the analysis of the results, following observations can be made

5.4.1 Effects of varying precedence

Precedence constraints have a considerable impact on various performance measures

(i) Average machine utilization (AMU) :

For zero flexibility, AMU show approximately 30 - 35 % increase between zero and medium precedence levels and approximately 5 - 10 % increase between medium and high precedence levels for all dispatching rules except SIO which has a relatively uniform rate of increase Performance is rather low for SIO in case of medium and high precedence and for FIFO in case of zero precedence WINQ was top scorer for medium precedence and EDD for zero precedence

When flexibility was set to medium, rate of increase became more uniform for all the dispatching rules AT rule offers the maximum AMU at medium and high precedence levels, the difference with other rules being more prominent at high precedence level At zero precedence level, all the rules perform more or less equally WINQ is the worst performer at medium precedence level

At high flexibility level, effect of changing precedence levels cause increase in AMU with a declining rate Rate of increase is highest in FIFO and lowest in WINQ At high precedence levels, FIFO offers best AMU and WINQ worst SIO is the best rule at zero precedence level

(ii) Mean flow time (MFT) :

The general trend in mean flow time is of 'increase' with varying precedence levels At zero flexibility, the rate of increase is more from zero to medium precedence for most

of the rules FIFO shows a linear increase whereas WINQ shows lower MFT at high precedence than at medium precedence SIO offers least MFT at all levels of precedence

At medium flexibility, FIFO shows a constant MFT at medium and high precedence So despite showing worst performance at medium precedence, it leads AT and EDD in terms of lowering MFT at high precedence Best performer is again SIO at medium and high levels of precedence All rules show approximately same MFT at zero precedence AT is the worst performer at all precedence levels

When flexibility is high, FIFO shows a linear increase in MFT with a very rate Thus it turns out to be worst performer with very high value of MFT at high precedence EDD stabilizes at medium precedence Other rules show declining rate of increase after medium precedence level EDD and SIO rules show better results at high precedence whereas WINQ and SIO show better results at medium precedence

(iii) Mean tardiness (MT) :

Mean tardiness also increases with increasing precedence level for all levels flexibility For zero flexibility case, the average percent increase from zero to medium precedence is 65-70% whereas from medium to high precedence , it is same for FIFO and AT, very low (1-2%) for WINQ, EDD, and SIO SIO rule is the best performer in reducing mean tardiness at medium and high precedence offering much lower values of MT than others

At medium flexibility, the rate of increase increases from medium to high precedence in case of WINQ, EDD, and AT rules For FIFO, it shows stability at medium and high precedence whereas SIO show a downward trend in rate of increase of MT However, SIO rule performs better than other rules at all levels of precedence The percent increase is abnormally high in cases of FIFO (85% from zero to medium precedence) and AT (55% from medium to high precedence)

At high levels of flexibility, MT for all rules increase with increasing precedence except EDD For EDD, it decreases from medium to high precedence FIFO shows a high and uniform rate of increase It offers MT nearly double the values offered by other rules at high precedence level For SIO, rate of increase declines after medium precedence

EDD rule is the best performer at high precedence level closely followed by SIO For medium precedence, WINQ and SIO are better performers and AT at zero precedence

(iv) Mean waiting time (MWT) :

Mean waiting time by different dispatching rules follow the same trend with varying precedence levels at various flexibility levels as that in case of MFT At zero flexibility level, SIO rule performs best at medium and high levels of precedence WINQ shows a drop in MWT from medium to high precedence (25%) EDD shows stable value of MWT after medium precedence

At medium flexibility, FIFO shows stable value of MWT after medium precedence Rate of increase increases by 10% from medium to high precedence in case of AT bringing its MWT value to highest position at high precedence WINQ show a linear increase SIO is again the best performer at medium and high precedence levels

When level of flexibility is high, FIFO shows a uniform and very high rate of increase in MWT Thus at high precedence, its MWT value is about 50% more than those of other rules Value of MWT decreases after medium precedence in case of EDD Other rules show linear increase EDD offers least MWT at high precedence, though it shows maximum at medium precedence SIO is the best rule at medium precedence level

5.4.2 Effects of varying flexibility

Flexibility produce considerable impact on various performance measures All the performance measures are discussed one by one with varying levels of flexibility at fixed levels of precedence

(i) Average machine utilization (AMU) :

At zero level of precedence, AMU is almost same for all dispatching rules, i e changing flexibility doesn't have considerable effect on AMU It first increases and then decreases in case of FIFO , decreases for EDD , increases for SIO and remains constant for WINQ Rate of change is very low in all the dispatching rules

At medium precedence also, rate of change is low AMU decreases with increasing flexibility in case of WINQ, increases for FIFO, and first increases then decreases for AT

At high precedence level, FIFO shows a linear increase in AU With flexibility increasing from zero to medium, AMU increases for all rules but when flexibility increases from medium to high , it decreases for WINQ,EDD,AT, and increases for FIFO and SIO FIFO offers maximum machine utilization at high flexibility whereas AT is the best rule at zero and medium flexibility

(ii) Mean flow time (MFT) :

Generally, mean flow time show a downward trend with increasing flexibility at all precedence levels At zero precedence level, percent decrease in MFT from zero to medium flexibility is about 35% for all rules It drops to 10% when flexibility increases from medium to high level There is only marginal difference in MFT values by different rules at all flexibility levels

For medium precedence, percent decrease is about 80% in cases of EDD and WINQ from zero to medium flexibility after which it stabilizes for WINQ MFT again increases after medium flexibility for EDD FIFO has constant value for MFT at low and medium flexibilities SIO offers lowest MFT at all flexibility levels

At high precedence level, MFT by FIFO rule first decreases from zero to medium flexibility and then it increases from medium to high flexibility SIO has near constant value for MFT with varying flexibility For other rules, MFT decreases SIO is the best rule at zero and medium flexibility levels and EDD at high flexibility FIFO is worst at high flexibility and AT at low and medium flexibility

(iii) Mean tardiness (MT) :

Mean tardiness also decrease with increasing flexibility at all precedence levels At zero precedence level, mean tardiness decreases by about 55% from zero to medium flexibility and by 15% from medium to high flexibility levels All rules show approximately same value of MT for all flexibility levels

At medium flexibility level, FIFO shows slight increase in MT from low to medium flexibility after which it decreases at a high rate MT due to EDD, after decreasing from zero to medium flexibility, increases at a significant rate SIO provides the least value of MT throughout At high flexibility, SIO and WINQ have comparable values for MT

At high precedence level, MT due to FIFO first decreases and then increases at almost same rate WINQ also shows first slight increase and then fall in MT For EDD and AT, rate of decrease increases form medium to high flexibility levels SIO has almost constant and lowest values of MT throughout For high flexibility, EDD and SIO have lower values of MT

(iv) Mean waiting time (MWT) :

Results for mean waiting time show almost similar trend as those shown by MFT for varying flexibility at different precedence levels

5.5 DISCUSSION

From the results obtained, SIO rule comes out to be clear winner in terms of mean flow time, mean tardiness, and mean waiting time at various levels of flexibility and precedence The reason for such good performance by SIO can be attributed to its nature of favoring parts with smaller processing times So, in a way it speeds up motion in the system, each workstation producing maximum number of parts in a given time interval Thus overall productivity of the system increases and mean flow time decreases With the decrease of mean flow time, mean waiting time also decrease as waiting time is a part of flow time As the parts which finish all their operations have shorter flow times, their lateness also decrease Thus SIO performs well in reducing mean tardiness of the parts also FIFO rule performs poorly with this criteria because it does not take into account any characteristic of the parts

Average machine utilization show increasing trend with increasing precedence level Intuitive explanation could be that increasing precedence might mean increased input of parts in the system because batch size now depends on the number of unused predecessors in the unloading which could be much more than the batch size of the part types which have no predecessors i e one to six So total time for which machines are busy increases as they have to entertain more parts now Increasing flexibility does not produce significant changes in average machine utilization This can be because alternate machines for the same operation have approximately same processing times for the sample

problem Hence the total workload remains same though it gets more evenly distributed with increasing flexibility So average machine utilization is almost same At high precedence level , with more parts in the system, the slight difference in processing times by alternate machines produce magnified workload on machines Hence average machine utilization increases

It is observed that mean flow time, mean tardiness and mean waiting time increase with increasing precedence at fixed levels of flexibility for all rules generally Same intuitive explanation can be given With increased precedence, there are more parts in the system, thus more congestion Hence mean flow time and mean waiting time increase With more parts, the number of tardy jobs also increase Hence mean tardiness increases As flexibility increases, a part can go to more than one machine for processing It goes to that machine that has least workload Thus, its chances of getting processed early increase and it leaves the system early Hence waiting time of the part decreases thus resulting in lower mean flow and waiting times With lower flow times, there would be less lateness among the jobs which are tardy Hence mean tardiness also decreases

WINQ also shows good results for part related performance measures after SIO It acts on global information considering status of the next machine As WINQ loads that part which will go to the next machine with least workload, it essentially balances the workload on machines Hence mean flow time and mean waiting time decrease and consequently mean tardiness decreases

Table 5 1 Number of machines and corresponding number of FMSs

No. of machines	No. of FMSs	No. of machines	No. of FMSs
1	4	10	8
2	6	11	1
3	4	12	4
4	7	13	2
5	10	15	1
6	11	16	1
7	8	28	1
8	4	29	1
9	5	80	1

Table 5 2 Probability distribution of processing time multipliers

Part type	Factors and Probabilities			
1	Factor	0 9	0 97	1 1
	Probability	0 3	0 3	0 4
2	Factor	0 93	1 03	1 09
	Probability	0 4	0 3	0 3
3	Factor	0 9	1	1 1
	Probability	0 2	0 5	0 3
4	Factor	0 9	1 04	
	Probability	0 5	0 5	

Table 5 3 Job processing data for experiments of no flexibility case (Ex00, Ex01, and EX02)

opn. no.	1	2	3	4
job type	machine / opn. tim	machine / opn. tim	machine opn. tim	machine / opn. tim
1	1/6	3/4	2/5	4/7
2	3/4	2/7	4/4	
3	1/5	2/6	4/3	
4	2/6	1/8	3/4	4/4

Table 5 4 Job processing data for experiments of medium flexibility case (Ex10, Ex11, and EX12)

opn. no.	1	2	3	4
job type	machine opn. tim	machine opn. tim	machine / opn. tim	machine / opn. tim
1	1/6 3/5	3/4 4/5	2/5 1/5 3/4	4/7 2/7
2	3/4 4/5	2/7 3/6	4/4 1/5	
3	1/5 2/6	2/6 3/7	4/3 1/3	
4	2/6 1/6 3/5	1/8 3/7	3/4 2/5	4/4 1/4

Table 5 5 Job processing data for experiments of high flexibility case (Ex20, Ex21, and EX22)

opn. no.	1	2	3	4
job type	machine opn. tim	machine / opn. tim	machine opn. tim	machine opn. tim
1	1/6	3/4	2/5	4/7
	3/5	4/4	1/5	2/6
	2/5	1/3	3/6	3/6
	4/6	2/3	4/4	1/7
2	1/3	2/7	4/4	
	2/4	3/7	2/5	
	3/4	1/6	1/4	
	4/4	4/7	3/4	
3	1/5	2/5	4/3	
	4/6	3/6	3/3	
	2/5	1/6	1/4	
	3/5	4/6	2/4	
4	2/6	1/8	3/4	4/4
	1/6	2/8	2/4	1/4
	3/6	3/7	1/3	2/3
	4/5	4/7	4/4	3/4

Table 5 6 . Results for Ex00

Rule	Avg. m/c utization	Mean flow time	Mean tardiness	Mean waiting time
FIFO	34 07	53 59	19 74	36 06
WINQ	37 39	54 38	20 94	37 08
EDD	41 05	56 83	21 75	39 21
AT	37 19	51 66	17 4	34 22
SIO	37 61	56 68	21 78	38 88

Table 5 7 Results for Ex01

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	51 31	108 6	62 87	90 3
WINQ	52 64	123 17	57 94	104 95
EDD	50 02	137 81	76 88	119 37
AT	51 37	105 49	59 98	87 27
SIO	43 63	88 71	48 38	70 52

Table 5 8 Results for Ex02

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	52 5	155 16	100 68	136 9
WINQ	52 71	101 42	59 06	83 1
EDD	56 47	133 46	87 36	115 14
AT	56 98	160 52	106 55	142 2
SIO	48 63	90	50 42	71 75

Table 5 9 Results for Ex10

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting tim
FIFO	39 28	43 71	10 4	26
WINQ	37 03	42 77	9 72	24 88
EDD	38 77	42 66	9 31	24 76
AT	38 76	44 74	10 86	26 82
SIO	38 68	42 36	8 83	24 13

Table 5 10 Results for Ex11

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	53 57	106 44	66 75	87 86
WINQ	48 74	73 06	34 81	54 47
EDD	50 29	76 52	39 03	57 95
AT	55 61	82 26	43 28	63 7
SIO	52 97	70 77	32 48	52 19

Table 5 11 Results for Ex12

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	60 59	106 31	65 7	87 65
WINQ	59 75	107 21	67 44	88 5
EDD	62 08	115 81	73 33	97 17
AT	67 98	137 59	95 39	118 92
SIO	57 63	88 35	48 77	69 64

Table 5 12 Results for Ex20

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	37 63	37 91	7 49	20 71
WINQ	37 25	38 07	7 57	20 93
EDD	36 82	36 8	6 39	19 47
AT	36 6	35 67	5 34	18 4
SIO	40 11	38 4	7 34	20 94

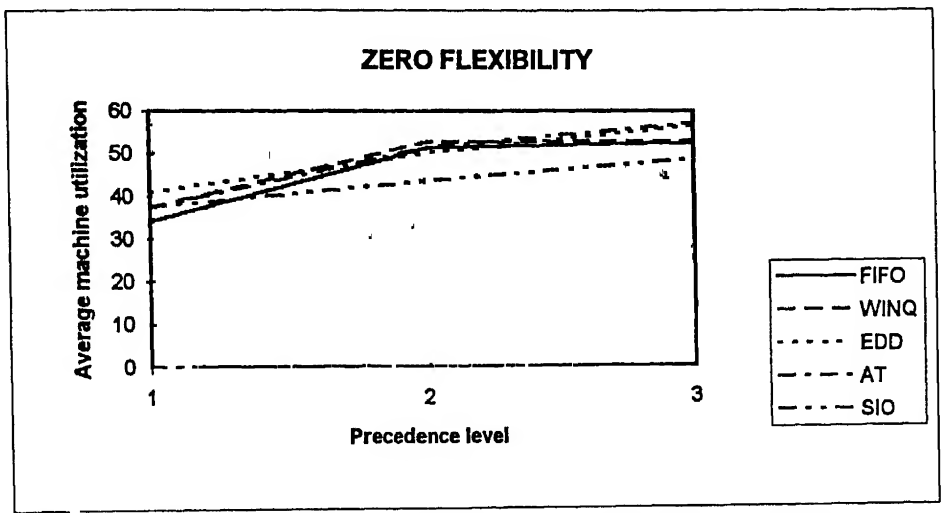
Table 5 13 Results for Ex21

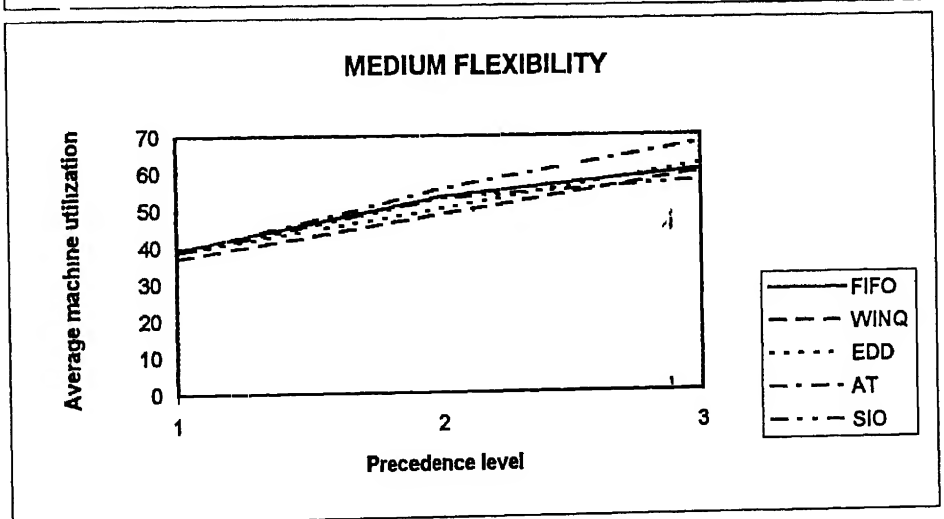
Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	54 3	86 42	49 42	68 72
WINQ	47 48	66 61	30 99	48 98
EDD	55 14	88 58	52 01	70 9
AT	54 36	71 17	34 27	53 58
SIO	52 82	66 15	30 4	48 44

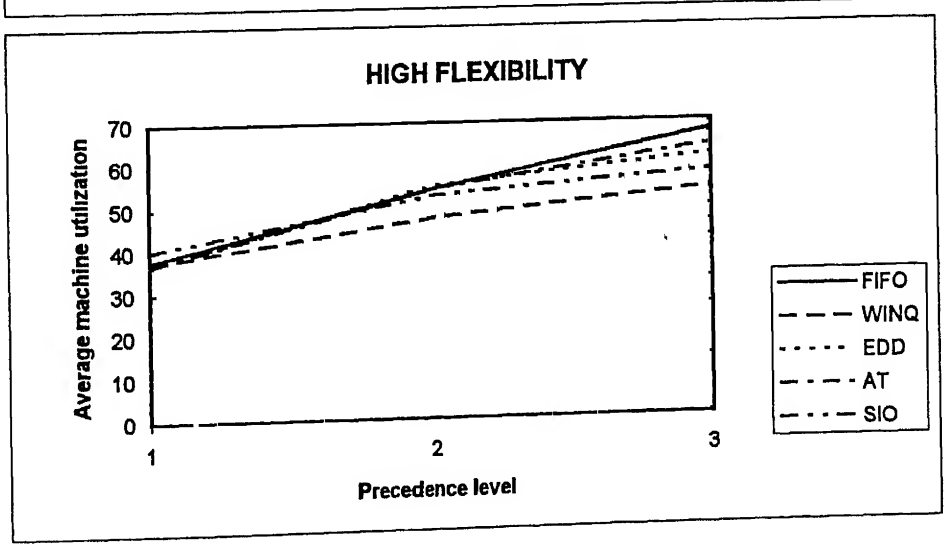
Table 5 14 Results for Ex22

Rule	Avg. m/c utization	Mean flow time	Mean tardines	Mean waiting time
FIFO	67 68	130 79	87 71	113 16
WINQ	53 97	91 98	54 35	74 26
EDD	61 84	85 58	47 56	67 86
AT	64 23	94 64	53 66	76 9
SIO	58 09	87 99	49 65	70 28

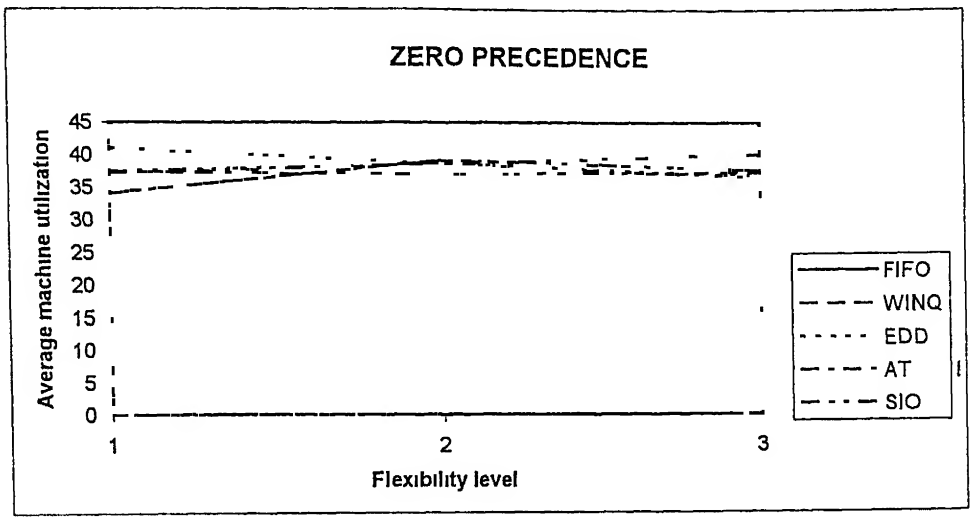
FIGURES 5.1, 5.2, 5.3 : EFFECTS OF VARYING PRECEDENCE ON AVERAGE MACHINE UTILIZATION AT FIXED FLEXIBILITY LEVEL

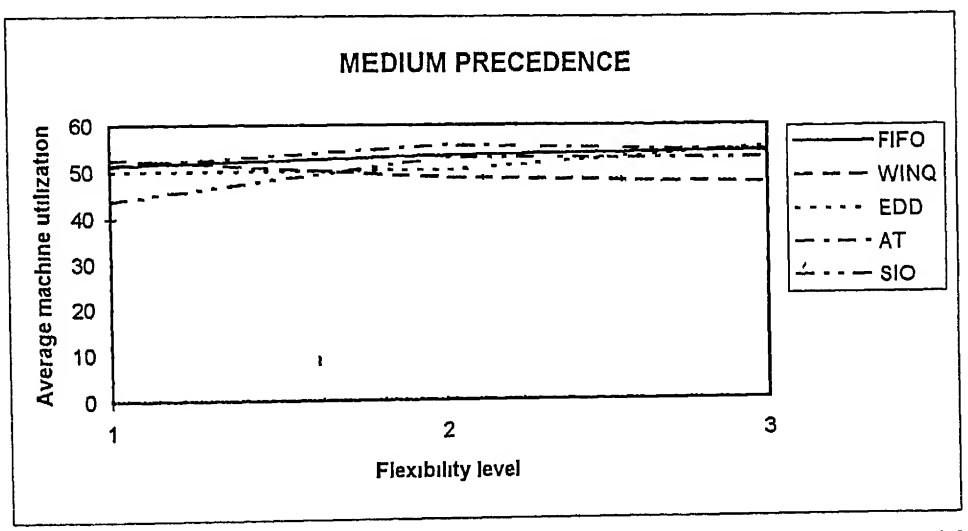


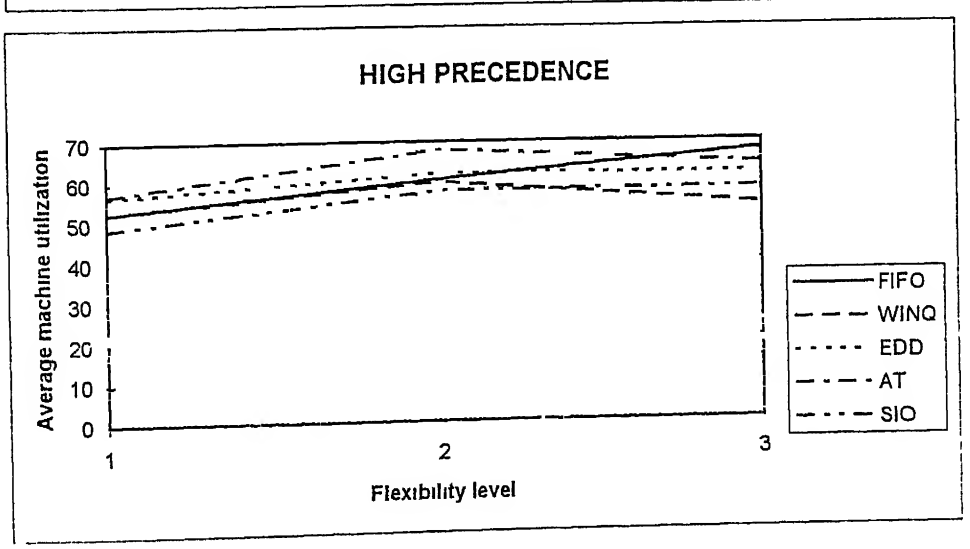




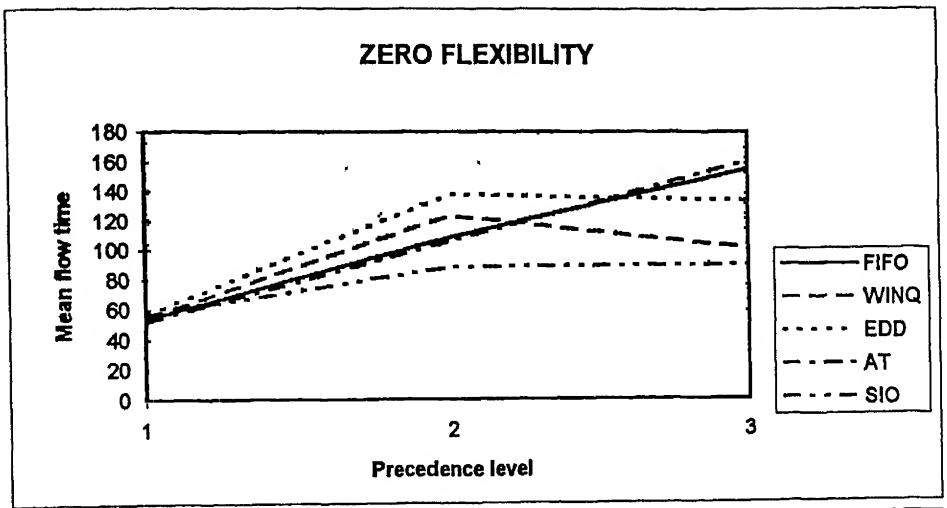
FIGURES 5 4, 5 5, 5 6 EFFECTS OF VARYING FLEXIBILITY ON AVERAGE MACHINE UTILIZATION AT FIXED PRECEDENCE LEVEL

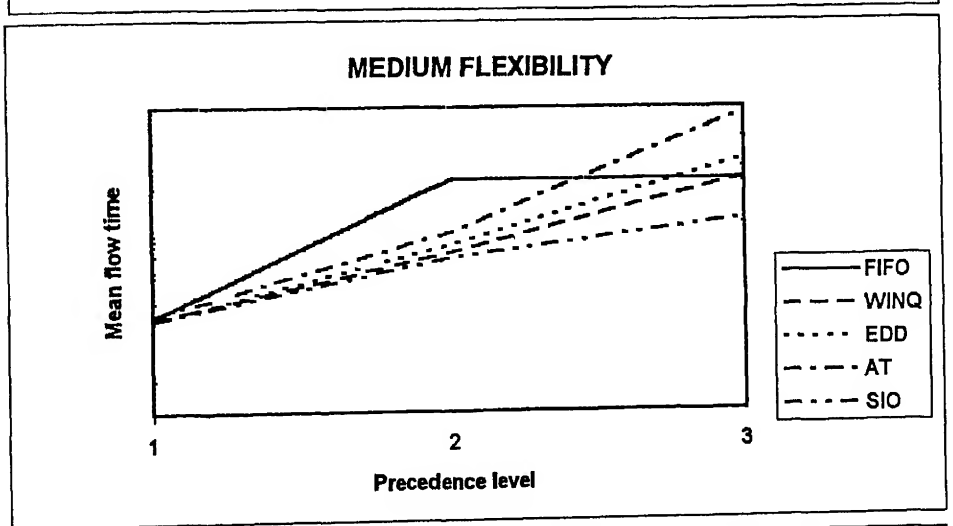


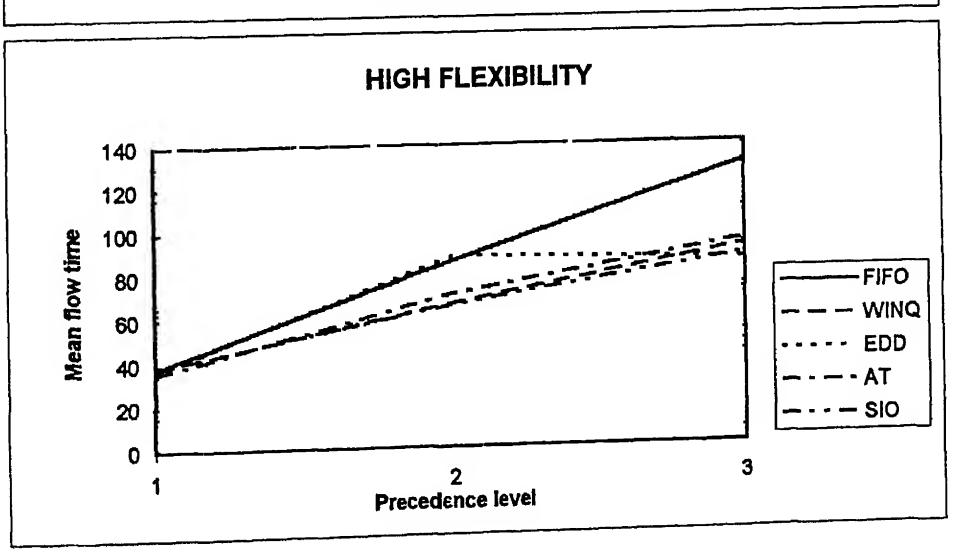




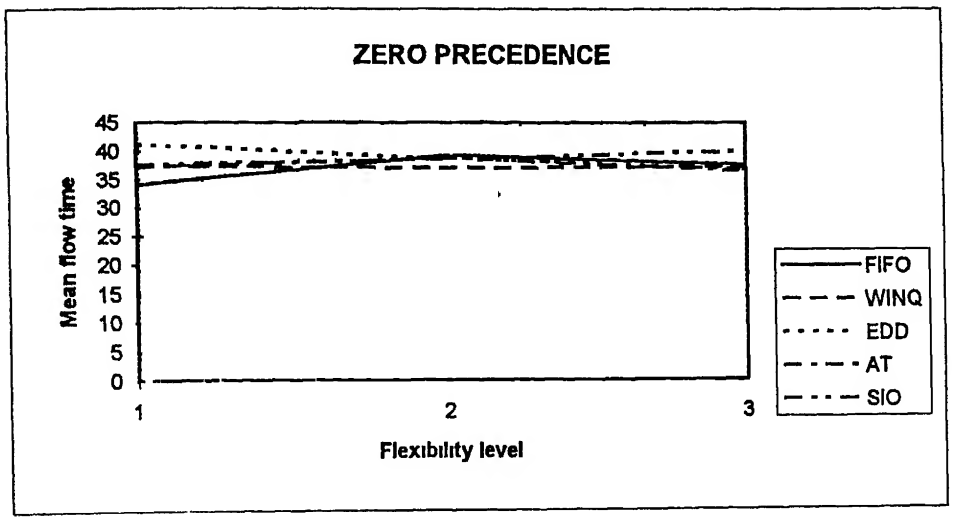
FIGURES 5.7, 5.8, 5.9 : EFFECTS OF VARYING PRECEDENCE ON MEAN FLOW TIME AT FIXED FLEXIBILITY LEVEL

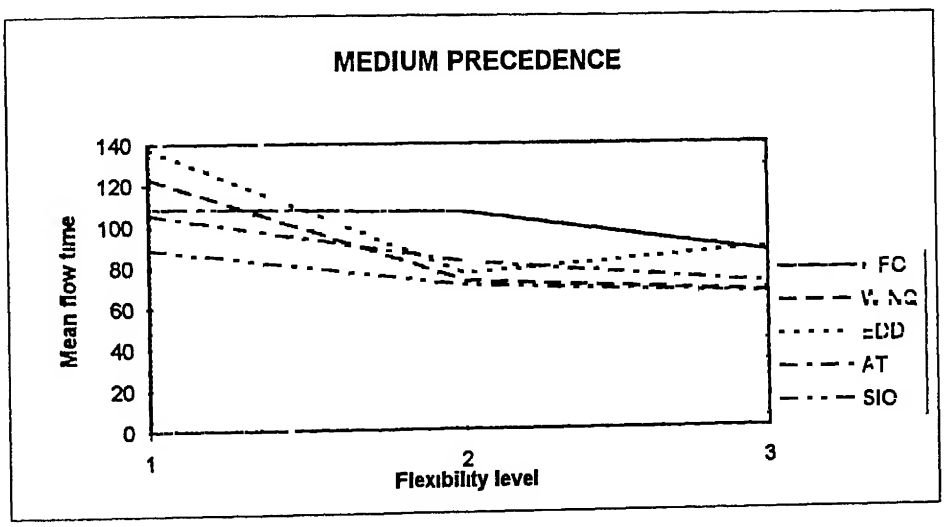


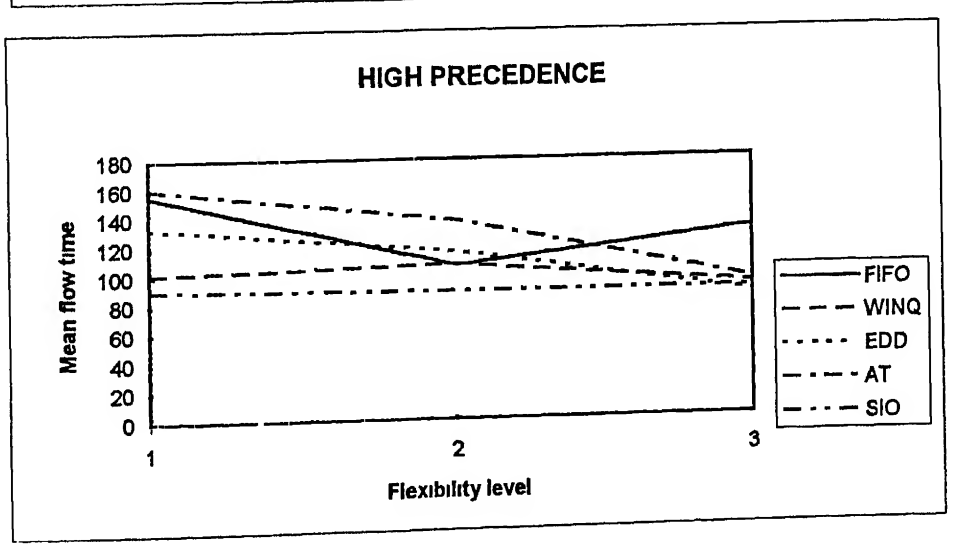




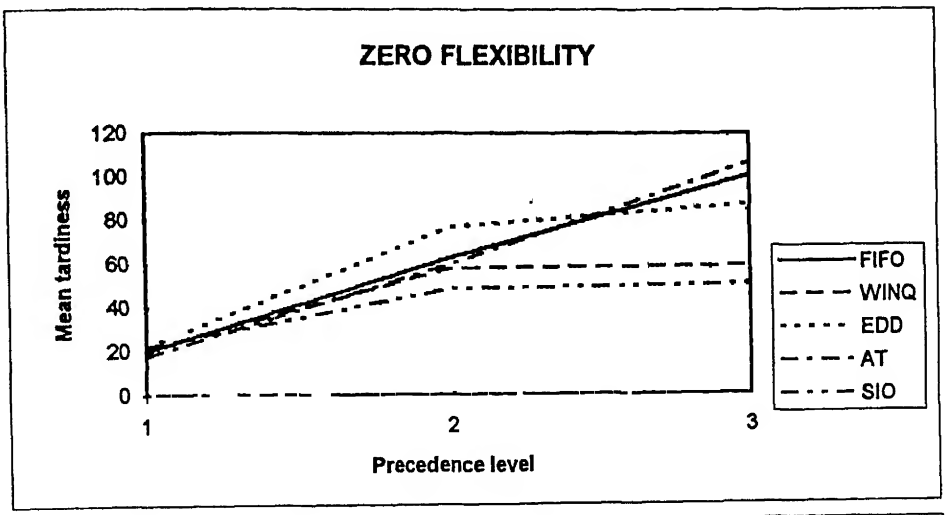
FIGURES 5.10, 5 11, 5 12 · EFFECTS OF VARYING FLEXIBILITY ON MEAN FLOW TIME AT FIXED PRECEDENCE LEVEL

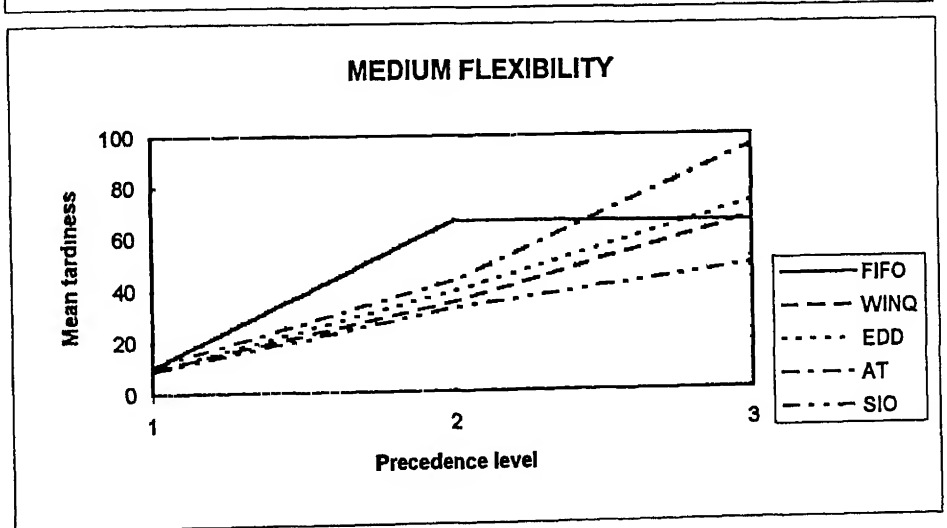


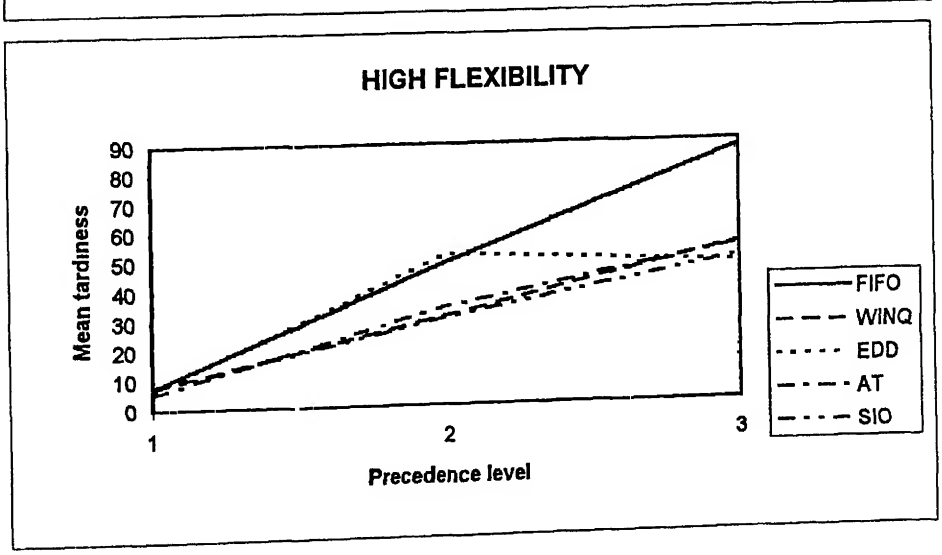




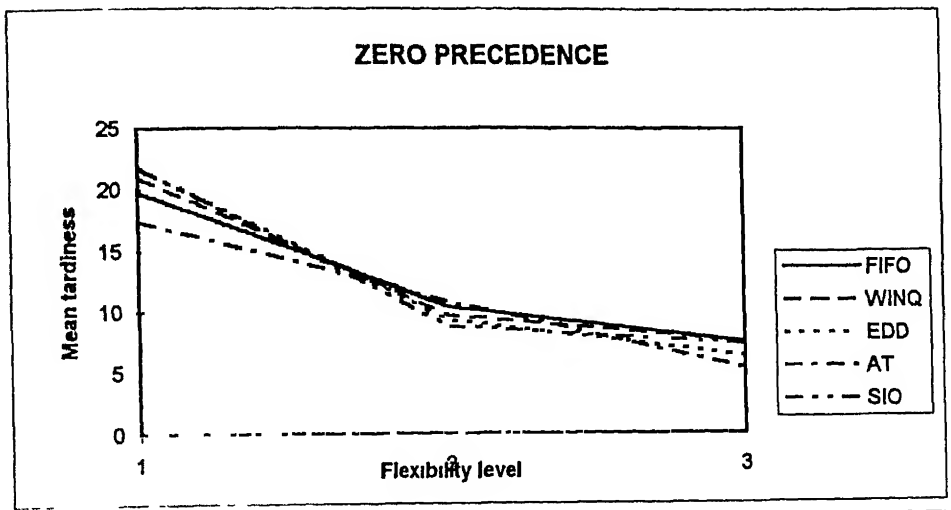
FIGURES 5.13, 5.14, 5 15 : EFFECTS OF VARYING PRECEDENCE ON MEAN TARDINESS AT FIXED FLEXIBILITY LEVEL

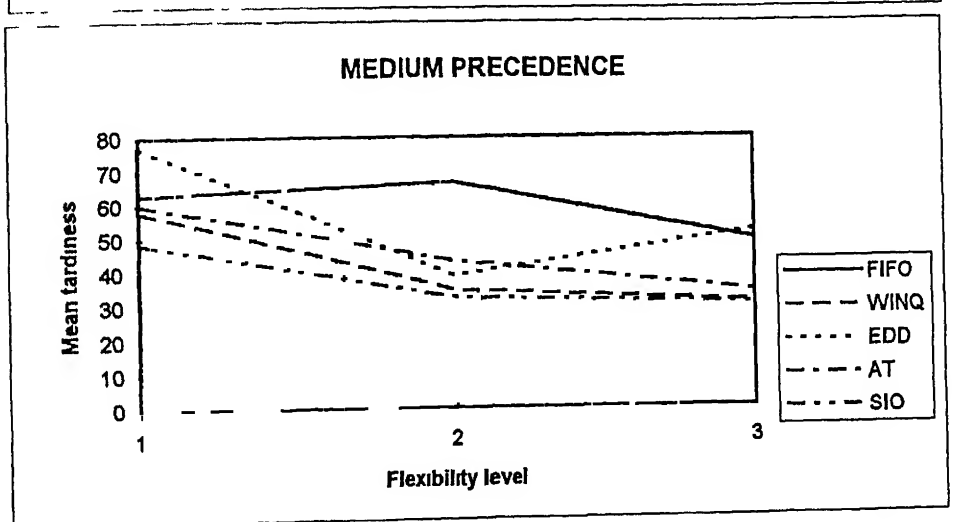


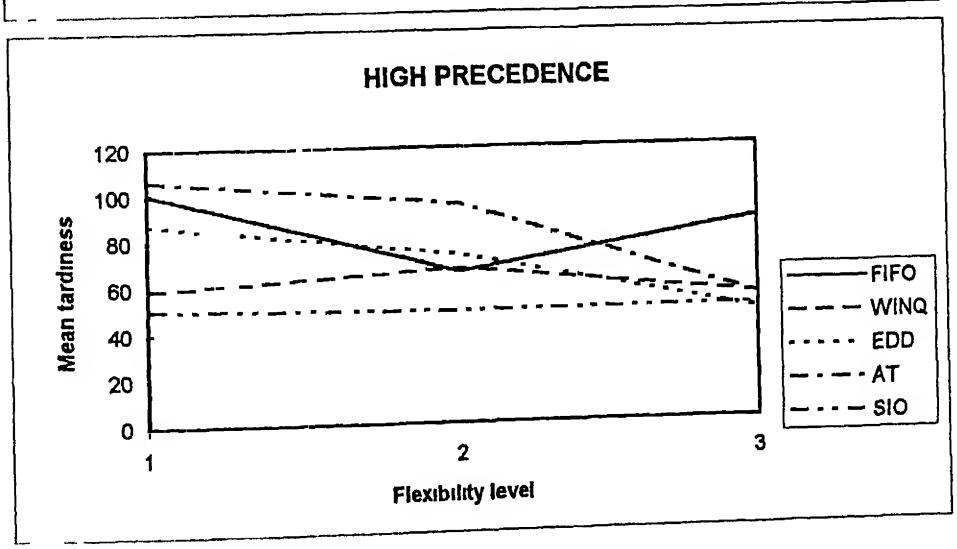




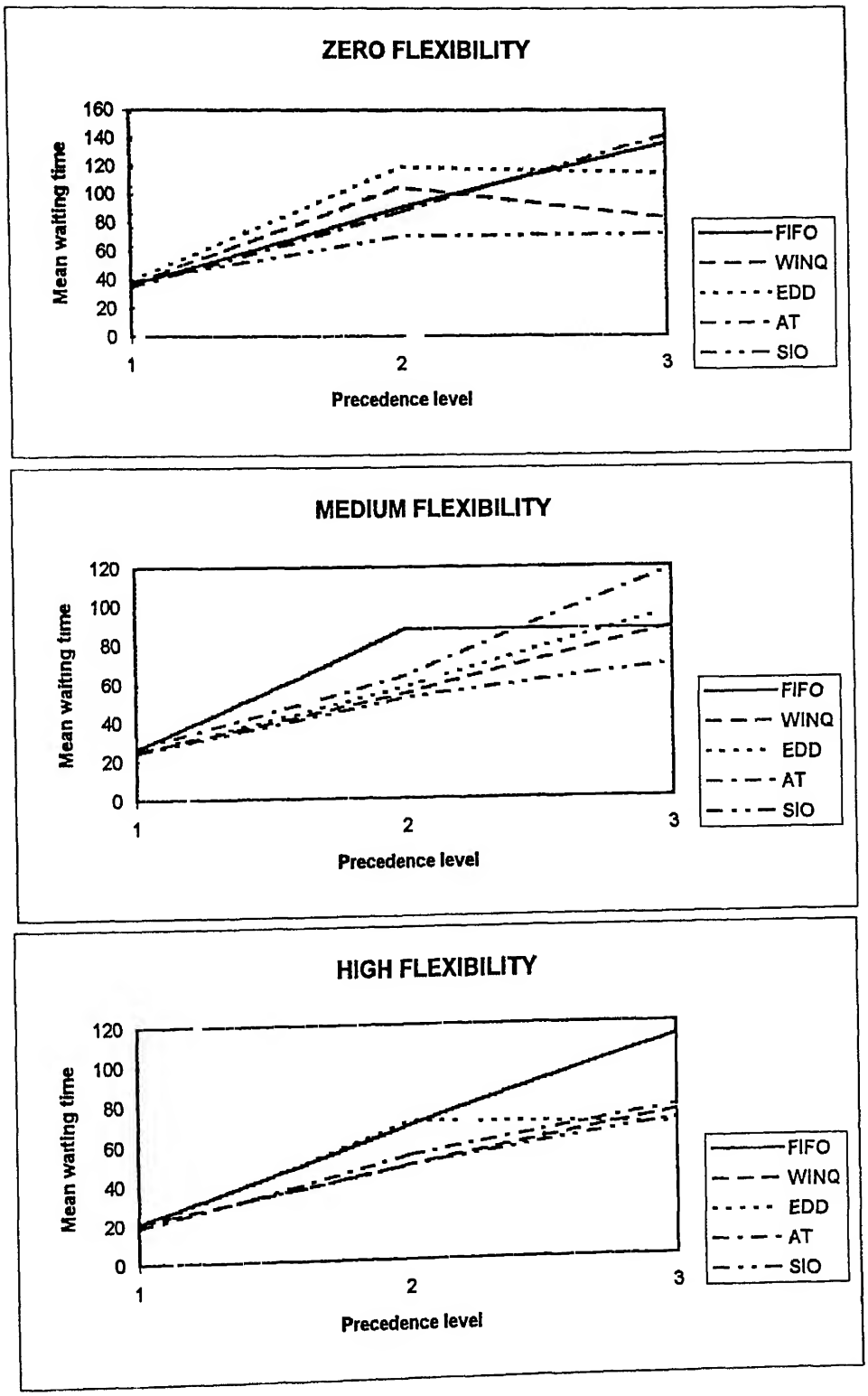
FIGURES 5.16, 5.17, 5.18 : EFFECTS OF VARYING FLEXIBILITY ON MEAN TARDINESS AT FIXED PRECEDENCE LEVEL



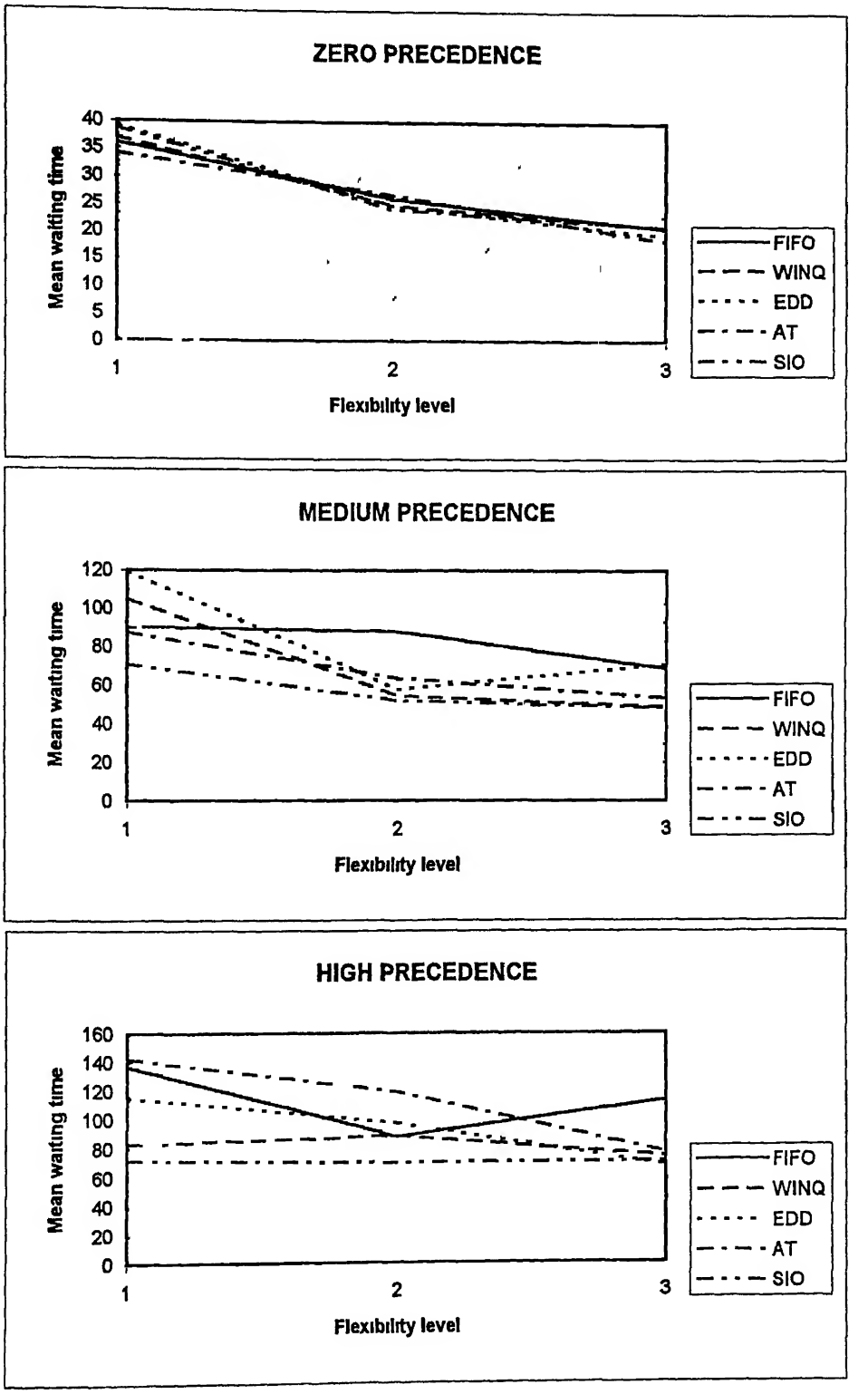




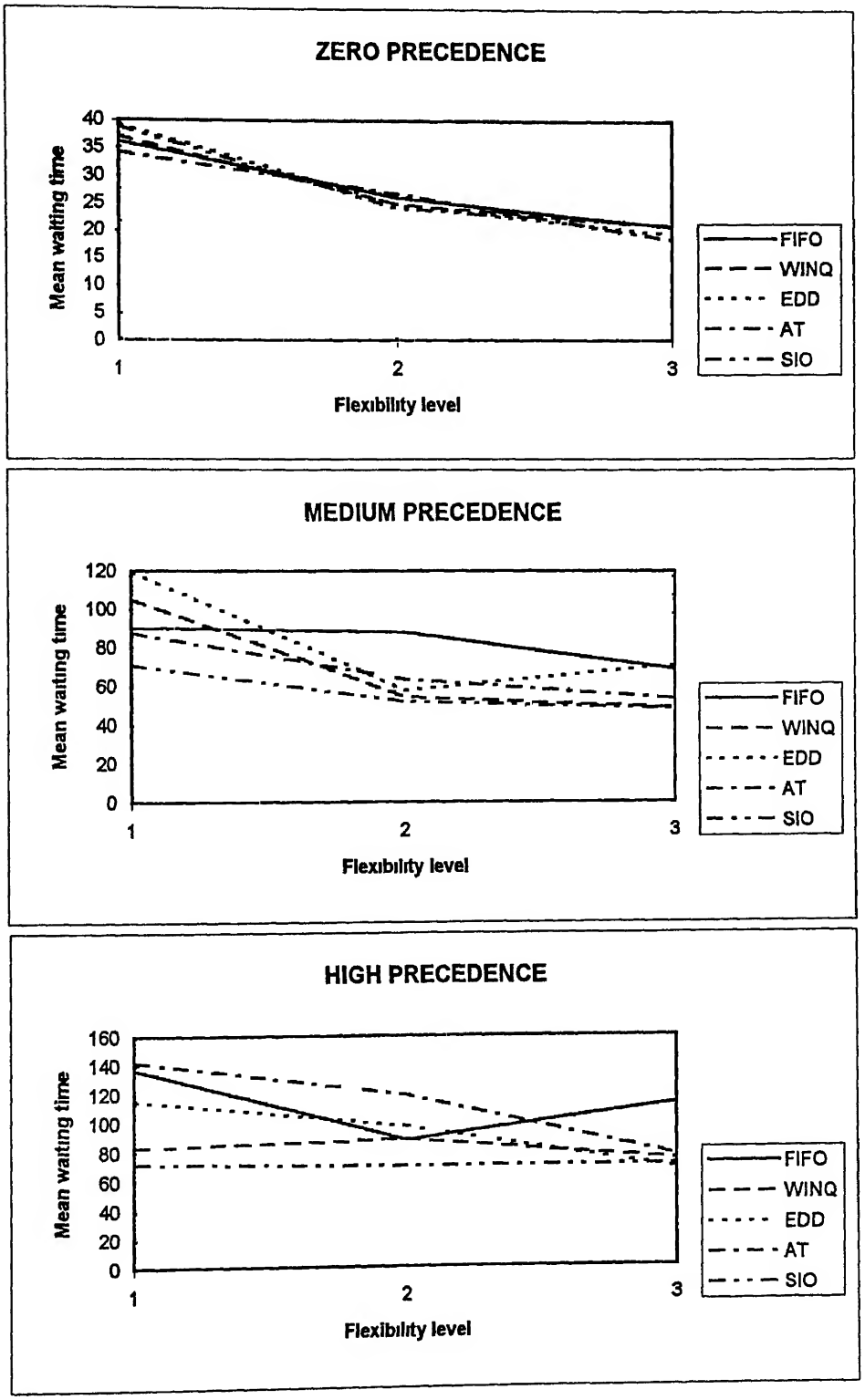
FIGURES 5.19, 5 20, 5.21 : EFFECTS OF VARYING PRECEDENCE ON MEAN WAITING TIME AT FIXED FLEXIBILITY LEVEL



FIGURES 5 22, 5.23, 5.24 : EFFECTS OF VARYING FLEXIBILITY ON MEAN WAITING TIME AT FIXED PRECEDENCE LEVEL



FIGURES 5 22, 5.23, 5.24 : EFFECTS OF VARYING FLEXIBILITY ON MEAN WAITING TIME AT FIXED PRECEDENCE LEVEL



CHAPTER SIX

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

The present work deals with the development of a comprehensive simulation model of FMS for testing of various dispatching rules, which are reported to be performing well in general job shop environment, under varying flexibility and precedence levels

It is generally felt that the modeling of an FMS for operational decision like dispatching is very complex However, it is further felt beyond any doubt that after having made several design level and other operational level decisions optimally, the success of an FMS greatly depends on employing a suitable dispatching rule The dispatching decision problem in FMS appears to be almost impossible to solve analytically Thus simulation can be considered as the only resort Various programming and statistical considerations need to be carefully paid attention to

The advantages offered by 'object oriented programming' philosophy in the area of manufacturing systems simulation appears to be very attractive Basic features of OOP viz abstraction, encapsulation, data hiding, inheritance, polymorphism, etc render software systems more understandable, modifiable, and reusable It is hoped that the present work paves the path for object oriented development models related to dispatching in FMS

The choice of 'C++' appears to be a good choice as object oriented programming language Its C ancestry brings C++ the tradition of an efficient, compact, fast, and portable language, and its object oriented heritage brings C++ a fresh programming methodology designed to cope with the escalating complexity of modern programming tasks

In the present system, in total, five dispatching rules are considered for comparison They are (i) FIFO, (ii) WINQ, (iii) EDD, (iv) AT, and (v) SIO The performance of the above mentioned dispatching rules for varying degrees of flexibility and precedence are considered From the limited experience of a sample problem,

1 Average machine utilization tend to increase with increasing precedence level at various degrees of flexibility It doesn't vary much when flexibility is increased at fixed level of precedence

2 All the part related performance measures viz mean flow time, mean tardiness, and mean waiting time generally increase with increasing levels of precedence at fixed flexibility and decrease with increasing flexibility at fixed precedence for all rules

3 SIO rule seems to have good potential in reducing the flow time and tardiness based objectives In 90% cases of varying flexibility and precedence, it outperformed other rules in terms of mean flow time, mean tardiness, and mean waiting time It is in accordance with most simulation findings from literature

4 For average machine utilization, performance of SIO is rather low FIFO seems to be a good rule as it indicates healthy rise in machine utilization with increasing precedence and flexibility levels AT also show high machine utilization at low and medium flexibility levels

5 FIFO shows poor performance for part related performance measures

6 Besides SIO, WINQ shows good results for mean tardiness, mean flow and waiting times at high levels of flexibility

7 Due date based rule, EDD, offer reduced tardiness at high precedence and flexibility levels

8 No single scheduling rule is found to be clear winner on all performance measures So it is up to the user to choose one or more of the scheduling rules according to the performance measures prevailing in his particular application

SCOPE FOR FURTHER WORK

The simulation model developed, can be further extended One of the key advantages of OOP is reusable software So using the classes developed for the present work, different configurations of the FMS can be modeled and studied with different objectives Following are some of the suggestions for further work in this area

1 As is clear from the discussion here, the dispatching rules behave somewhat differently in different flexibility and precedence conditions Some of these rules work well in no flexibility kind of environment and some of them may work in flexible kind of situation These qualities of various rules can be exploited in an effective manner for different kind of FMS configurations and flexibilities

2 Different dispatching rules, tailor-made for FMS environment can be compared

3 Besides routing flexibility, other types of flexibilities can also be introduced and their effect can be studied

4 Present work doesn't take into account, material handling system with detail So classes related to AGV can be developed and included in the model to give a more practical approach towards material handling

5 Interaction of various dispatching strategies in a shop can be analyzed to find a unified approach in scheduling

6 Most manufacturing firms are embracing the ideals of JIT manufacturing This requires a new approach to organize manufacturing operations Flexible manufacturing is a key technique for achieving the reduction in lead times and work in process levels which, the JIT method aims at One of the effects of this change is that manufacturing systems are now required to operate on a pull method rather than a push basis

This change in approach demands a rethink about simulation modeling of FMS Almost all the simulation models developed for FMS environment are based on the push principle To implement the JIT approach, we need a method of pulling work through the system The decision rules must be framed to suit this particular situation

REFERENCES

1 Adiga, S, and Glassey, C R, 1991, "Object-oriented simulation to support research in manufacturing systems", International Journal of Production Research, 29, pp 2529-2542

2 Anderson, E J, and Nyirenda, J C, 1990, "Two new rules to minimize tardiness in a job shop", International Journal of Production Research, 28(12), pp 2277-2292

3 Arizono, I, Yamamoto, A, and Hiroshiohta, 1992, " Scheduling for minimizing total actual flow time by neural networks ", International Journal of Production Research, 30(3), pp 513-526

4 Benjaafar, S, "Models for performance, evaluation of flexibility in manufacturing systems", International Journal of Production Research, 32(6), pp 1383-1402

5 Blackstone, J H Jr, Phillips, D T, and Hogg, G L, 1982, "A state-of the-art survey of dispatching rules for manufacturing job shop operations", International Journal of Production Research, 20(1), pp 27-45

6 Booch, G, 1991, Object-Oriented Design with Applications, (California Benjamin Cummings)

7 Cargill, T, 1992, C++ Programming Style, Addison Wesley Publishing Company

8 Chandra, J, and Talavage, J, 1991, "Intelligent dispatching for flexible manufacturing",International Journal of Production Research,29(11), pp 2259-2278

9 Co, H, Biermann, J S, and Chen, S K, 1990, "A methodical approach to the FMS batching, loading, and tool configuration problems", International Journal of Production Research, 28(12), pp 2171-2186

10 Denzler, D R, Boe, W J, 1981, "Experimental investigation of FMS scheduling problem", International Journal of Production Research, 25(7), pp 979-994

11 Gupta, Y P, and Goyal, S, 1992, "Flexibility tradeoffs in a random FMS",

12 Hwang, S S, and Shogan, A W, 1989, "Modeling and solving an FMS part selection problem",International Journal of Production Research, 27(8), pp 1349-1366

13 Kim, Y D, and Jano, C A, 1994, "A due date based approach to part type selection in FMSs", International Journal of Production Research, 32(5), pp 1027-1043

14 Kim, Y D, 1993, "A study on surrogate objectives for loading a certain type of FMSs ", International Journal of Production Research, 31(2), pp 381-392

15 Kimemia, J, and Stanley, B G, 1985, "Flow optimization in FMS", International Journal of Production Research, 23, pp 81-96

16 Kusiak, A, 1985, "Flexible manufacturing systems A structural approach", International Journal of Production Research, 31, pp 1057-1073

17 Law, A M, and Kelton, W D, 1991, Simulation Modeling and Analysis, Mc-Graw Hill

18 MacCarthy, B L, and Liu, J, 1993, " A new classification scheme for FMS", International Journal of Production Research, 31, pp 299-309

19 Modi, B K, and Shanker, K, 1994, "Models and solution approaches for part movement minimization in FMS with machine tool and process plan flexibilities", International Journal of Production Research, 33(7), pp 1791-1816

20 Mohamed, Z M, and Bernardo, J J, 1995, "The relationship between part type demand and FMS loading policies", International Journal of Production Research, 33(8), pp 2197-2211

21 Montazeri, M, and Wassenhove, L N, 1990, "Analysis of scheduling rules for an FMS", International Journal of Production Research, 28(4), pp 785-802

22 Mukhopadhyay, S K, Midha, S, and Krisna, Y A, 1992, "A heuristic procedure for loading problems in FMS", International Journal of Production Research, 30(9), pp 2213-2228

23 Mukhopadhyay, S K , Maiti, B , and Garg, S , 1991, "Heuristic solution to the scheduling problems in FMS", International Journal of Production Research, 29(10), pp 2003-2024

24 Nandkeyolyar, U , and Christy, D P , 1992, "An investigation of the effect of machine flexibility and number of part families on system performance", International Journal of Production Research,30(3), pp 513-526

25 O'Grady, P J , and Menon, U , 1987, "Loading an FMS", International Journal of Production Research, 25, pp 1053-1068

26 Prata, S , 1996, C++ Primer Plus, Galgotia Publications Private Limited, New Delhi

27 Raghu, T S , and Rajendran, C , 1995, "Due date setting methodologies based on simulated annealing", International Journal of Production Research, 33(9), pp 2535-2554

28 Scultz, C R , 1989, "An expediting heuristic for the SPT heuristic rule ", International Journal of Production Research, 27, pp 31-41

29 Shanker, K , and Tzen, Y J J , 1985, "A loading and dispatching problem in a random flexible manufacturing system", International Journal of Production Research, 23, pp 579-595

30 Shanker, K , and Srinivasulu, A , 1989, "Some solution methodologies for loading problems in FMS", International Journal of Production Research, 27(6), pp 1019-1034

31 Stecke, K E , and Solberg, J J , 1981, "Loading and control policies for an FMS", International Journal of Production Research, 29, pp 273-281

32 Stoeva, S P , 1990, " A due date based dispatching rule for FMS", International Journal of Production Research, 28, pp 1991-2000

33 Stroustrop, B , 1986, The C++ Programming Language, Addison Wesley Publishing Company

34 Woo-Tsong, L , and Chang, J C , 1995, "An object-oriented analysis approach in

BIBLIOGRAPHY

1 Adıga, S , 1989, "Software modelıng of manufacturıng systems a case for an object-orıented modelıng approach", Annals of Operatıons Research, 17, pp 363-378

2 Baker, K R , 1984, "Sequencıng rules and due date assignments ın a job shop", Management Scıence, 30, pp 1093-1104

3 Bernardo, J J , and Mohamed, Z M , 1992, "The measurement and use of operatıonal flexıbılıty in the loadıng of flexıble manufacturıng systems ", European Journal of Operatıonal Research, 60(2), pp 144-155

4 Berrada, M , and Stecke, K E , 1986, "A branch and bound approach for machıne loadıng ın flexıble manufacturıng systems", Management Scıence, 32(10), pp 1316-1335

5 Browne, J , Dubois, D , Rathmull, K , Sethı, S P , and Stecke, K E , 1984, "Classıfıcatıon of flexible manufacturıng systems", The FMS Magazıne, pp 114-117

6 Buzacott, J A , and Mandelbaum, m , 1985, "Flexıbılıty and productıvıty ın manufacturıng systems", Proceedıngs of the annual IIE Conference, Los Angeles, CA, pp 404-413

7 Byrkett, D L , Ozden, M H , and Patton, J M , 1988, " Intelligent FMS wıth tradıtıonal manufacturıng, plannıng, and control", Production and Inventory Management Journal, 29, pp 15-21

8 Cacı, 1983, SIMSCRIPT 11 5 Programmıng Language (Los Angeles Cacı, Inc)

9 Carter, M F , 1986, "Desıgnıng flexıbılıty ınto automated manufacturıng systems", Proceedıngs of the 2nd ORSA/TIMS Conference on FMS, pp 107-118

10 Chakravarty, A K , and Shtub, A , 1984, "Selectıng parts and loadıng flexıble manufacturıng systems", Proceedıngs of the 1st ORSA/TIMS Conference on FMS, pp 284-289

11 Chatterjee, A , Cohen, M , Maxwell, W , and Hiller, L , 1984, "Manufacturing flexibility Models and Measurements", Proceedings of the 1st ORSA/TIMS Special Interest Conference on FMS, Ann Arbor, MI, pp 49-64

12 Conway, R W , 1965, "Priority dispatching and job lateness in a job shop", The Journal of Industrial Engineering, 16(4), pp 228-237

13 Cox, B , 1986, Object-oriented programming An evolutionary approach, Addison Wesley, Reading, MA

14 Deckro, R , Herbert, J , and Winkonfsky, E , 1982, "Multiple criteria job shop scheduling", Computers and Operation Research, Vol 9, No 4, pp 279-285

15 Dupont, and Gatelmand, K , 1981, "A survey of flexible manufacturing systems", Journal of Manufacturing Systems, 1, 1

16 Ghiassi, M , DeVor, R , Dessouky, M , and Kijowsky, B , 1984, "An application of multiple criteria decision making principles for planning operations", IIE Transactions, Vol 16, No 2, pp 106-114

17 Grant, Hank, F , 1989, "Tutorial scheduling manufacturing systems with FACTOR " Proceedings of the IX ICPR, Manitoba, Canada, pp 277-280

18 Greene, T J , and Sadowski, R , 1986, "A mixed integer program for loading and scheduling multiple manufacturing cells", European Journal of Operational Research, 24, pp 379-386

19 Irastorza, J C , and Deane, R H , 1974, "A loading and balancing methodology for job shop control", AIIE Transactions, 6, pp 302-309

20 Jaikumar, R , 1986, "Post Industrial Manufacturing", Harvard Business Review, 64(6), pp 69-76

21 Jones, C V , Maxwell, W L , 1986, "A system for manufacturing scheduling with interactive computer graphics", IIE Transactions, pp 298-303

22 Kaltwasser, J , Hercht, A , and Lang, R , 1986, "Hierarchical control of FMS", IFAC Information Control Problems in Manufacturing Technology, Suzdal, USSR,

23 King, Christina, U , and Fisher, E L , 1986, "Object-oriented design, simulation, and evaluation ", Proceedings of 1986 Full Industrial Engineering Conference, pp 131-137

24 Kiran, A S , and Tansel, B C , 1986, "The system setup in FMS concepts and formulations", Proceedings of the 2nd ORSA/TIMS Conference on FMS, pp 321-332

25 Klein, B , 1986, "SIMFACTORY tutorial " Proceedings of the 1986 Winter Simulation Conference, pp 530-542

26 Larkin, T S , Carruthers, R I , and Soper, R S , 1988, "Simulation and object-oriented programming the development of SERB " Simulation, 52(3), pp 93-100

27 Law, A M , 1988, "Simulation of manuacturing systems", Proceedings of the 1988 Winter Simulation Conference, San Diego, CA, pp 40-51

28 Lomow, Greg, and Baezner, Dirk, 1989, "A tutorial introduction to object-oriented simulation ans sim++ ", Proceedings of the 1989 Winter Simulation Conference, pp 140-146

29 Mascarenhas, B , 1981, "Planning for flexibility", Long Range Planning, 14(5), pp 78-82

30 Meyer, B , 1987, "Reusability The case of object-oriented design", IEEE Software, 4, 2, pp 50-64

31 Mize, J H , Bhuskute, H C , Pratt, D B , and Kamath, M , 1992, "Modeling of integrated manufacturing systems using an object-oriented approach", IIE Transactions, pp 14-25

32 Nazmi, Adeel, and Stein, Susan, J , 1989, "Comparison of conventional and object-oriented approaches for simulaton of manufacturing systems " Proceedings of 1989 IIE Integrated Systems Conference, pp 471-476

33 Nof, S Y , Barash, B M , and Solberg, J J , 1979, "Operational control of item flow in versatile manufacturing systems", International Journal of Production Research, 17, pp 479-491

34 Norbis, M I, and Smith, J M, 1988, "A multiobjective, multi-level heuristic for dynamic resource constrained scheduling problems", European Journal of Operation Reseach, Vol 33, pp 30-41

35 Pegden, C, 1985, "Introduction to SIMAN" ,(Systems Modeling Corporation, PA)

36 Pritsker, A A P, and Pegden, C, 1979, Introduction to simulation and SLAM, (Halstead Press, John Wiley & Sons)

37 Rajagopalan, S, 1986, "Formulation and heuristic solutions for part grouping and tool loading in an FMS",Proceedings of the 2nd ORSA/TIMS Conference on FMS, pp 311-320

38 Schreiber, T J, 1974, Simulation using GPSS (New York Wiley)

39 Smith, M L, Ramesh, R, Dudek, R A, and Blair, E L, 1986, "Characteristics of U S flexible manufacturing systems - a survey" Proceedings of the 2nd ORSA/TIMS Conference on FMS OR Models and Applications, Ann Arbor, MI, pp 477-486

40 Stecke, K E, 1981, "Production planning for FMS", PhD thesis, School of Industrial Engineering, Purdue University, Indiana

41 Stecke, K E, 1983, "Formulation and solution of nonlinear integer production planning problems for FMS", Management Science, 29(3), pp 273-288

42 Stecke, K E, and Morin, T L, 1985, "The optimality of balancing workloads in certain types of FMS", European Journal of Operational Research, 20, pp 68-82

43 Stecke, K E, and Solberg, J J, 1985, "The optimality of unbalancing both work-loads and machine group sizes in closed queueing networks of multiple server queues", Operations Research, 33, pp 882-910

44 Warnecke, H J, and Steinhilper, R, 1982, "FMS new concepts, edp-supported planning, application examples", Proceedings of the 1st International Conference on FMS, Brighton, UK, pp 345-357

45 Witney, C K, and Goul, T S, 1980, "Sequential decision procedures for batching balancing in FMS", Annals of Operations Research, 3, pp 301-316

34 Norbis, M I , and Smith, J M , 1988, "A multiobjective, multi-level heuristic for dynamic resource constrained scheduling problems", European Journal of Operation Reseach, Vol 33, pp 30-41

35 Pegden, C , 1985, "Introduction to SIMAN" ,(Systems Modeling Corporation, PA)

36 Pritsker, A A P , and Pegden, C , 1979, Introduction to simulation and SLAM, (Halstead Press, John Wiley & Sons)

37 Rajagopalan, S , 1986, "Formulation and heuristic solutions for part grouping and tool loading in an FMS",Proceedings of the 2nd ORSA/TIMS Conference on FMS, pp 311-320

38 Schreiber, T J , 1974, Simulation using GPSS (New York Wiley)

39 Smith, M L , Ramesh, R , Dudek, R A , and Blair, E L , 1986, "Characteristics of U S flexible manufacturing systems - a survey " Proceedings of the 2nd ORSA/TIMS Conference on FMS OR Models and Applications, Ann Arbor, MI, pp 477-486

40 Stecke, K E , 1981, "Production planning for FMS", PhD thesis, School of Industrial Engineering, Purdue University, Indiana

41 Stecke, K E , 1983, "Formulation and solution of nonlinear integer production planning problems for FMS", Management Science, 29(3), pp 273-288

42. Stecke, K E , and Morin, T L , 1985, "The optimality of balancing workloads in certain types of FMS", European Journal of Operational Research, 20, pp 68-82

43 Stecke, K E , and Solberg, J J , 1985, "The optimality of unbalancing both work-loads and machine group sizes in closed queueing networks of multiple server queues", Operations Research, 33, pp 882-910

44 Warnecke, H J , and Steinhilper, R , 1982, "FMS new concepts, edp-supported planning, application examples", Proceedings of the 1st International Conference on FMS, Brighton, UK, pp 345-357

45 Witney, C K , and Goul, T S , 1980, "Sequential decision procedures for batching balancing in FMS", Annals of Operations Research, 3, pp 301-316